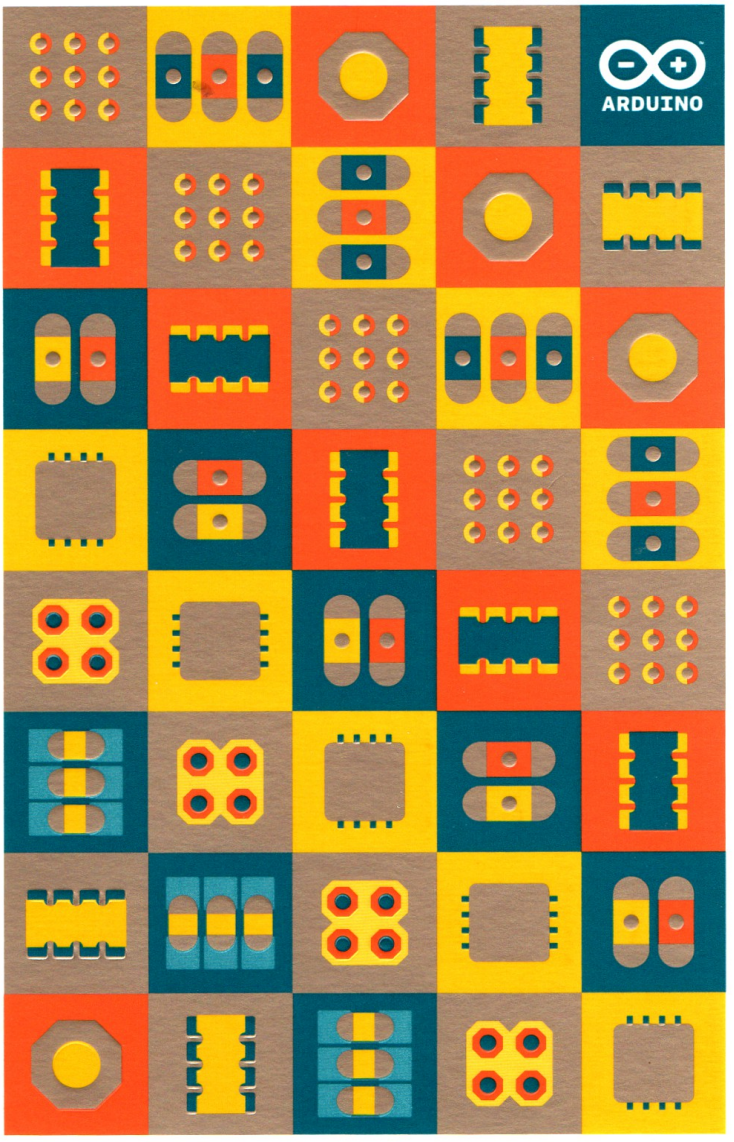
ARDUINO LIBRO DE PROYECTOS



ARDUINO LIBRO DE PROYECTOS

Lo autores del libro de proyectos de Arduino en inglés "ARDUINO PROJECTS BOOK":
Proyectos y textos por Scott Fitzgerald y Michael Shiloh
Revisión del texto por Tom Igoe

DISEÑO Y DIRECCIÓN DE ARTE
TODO
Giogio Olivero, Mario Ciardulli, Vanessa Poli, Michelle Nebiolo
todo.to.it

FABRICACIÓN DIGITAL Y GESTIÓN DE PROYECTOS
Oficina de Arduino en Torino
Katia De Coi, Enrico Bassi

ASESORES Y APOYO
Massimo Banzi, Gianluca Martino, Smart Projects

PROBADORES DE PROYECTOS Y CORRECTORES
Michael Shiloh, Michelle Nebiolo, Katia De Coi, Alessandro Buat, Federico Vanzati, David Mellis, Luisa Castiglioni

GRACIAS
Muchas gracias a toda la comunidad de Arduino por su continua contribución, soporte y sugerencias.
También gracias en especial al equipo de Fritzing: algunas de las ilustraciones de los componentes electrónicos utilizadas en este libro son tomadas o modificadas del proyecto Open-Source de Fritzing (www.fritzing.org)
Gracias de corazón a Paul Badger por la librería "CapacitveSensor" usada en el Proyecto número 13

El texto del Libro de Proyectos de Arduino está bajo una Licencia Creative Commons Reconocimiento – NoComercial -CompartirIgual 3.0 del 2012 por Arduino LLC. Esto quiere decir que puede copiar, volver a usar, y basarse en el texto de este libro con fines no comerciales e indicar la autoría del trabajo original (pero no de una manera que sugiera que estamos de acuerdo con usted o con el uso de la obra) y solo si los resultados están realizados bajo la misma Licencia Creative Commos.
Términos completos de la licencia:
creativecommons.org/licenses/by-nc-sa/3.0/es/

2012 – 2013 Arduino LLC. El nombre y el logotipo de Arduino son marcas de Arduino, registradas en U.S.A y en el resto del mundo. La mención de otros productos y nombres de compañías dentro de este libro también son marcas de sus respectivas compañías

La información en este libro esta distribuida como un "Como está" básico sin garantías adicionales. Si bien todas las precauciones se han tenido en cuenta en el diseño de este libro, ni los autores ni Arduino LLC tienen ninguna responsabilidad debido al daño o el perjuicio producidos por cualquier persona o entidad en el momento de llevar a cabo las instrucciones indicadas en este libro o por la utilización del software y el hardware descritos en él.

Este libro no se puede vender por separado del Kit de Inicio de Arduino.

Diseñado, impreso y encuadernado en Torino, Italia.
Septiembre 2012

Segunda edición, Mayo 2013

TRADUCCIÓN
Autor: Florentino Blas Fernández Cueto (Tino Fernández)
Sitio web: http://www.futureworkss.com
Bajo una Licencia Creative Commons Reconocimiento – NoComercial -CompartirIgual 3.0 del 2015 por futureworkss

Traducción revisada por:
Miguel Carlos de Castro Miguel, Ingeniero Químico y Profesor de Tecnología
Manuel Díaz Santalla, Técnico Superior de Electricidad

ÍNDICE

4	00 INTRODUCCIÓN
20	01 Conozca sus herramientas
32	02 Interface de nave espacial
42	03 Medidor de enamoramiento
52	04 Lámpara de mezcla de colores
62	05 Indicador del estado de ánimo
70	06 Theremin controlado por luz
78	07 Teclado musical
86	08 Reloj de arena digital
94	09 Rueda de colores motorizada
102	10 Zoótropo
114	11 La Bola de Cristal
124	12 Mecanismo de bloqueo secreto
136	13 Lámpara sensible al tacto
144	14 Retocar el logotipo de Arduino
156	15 Hackear botones
162	A/Z Glosario
170	Apuntes y libros recomendados

00

Todo el mundo, cada día, utiliza la tecnología. La mayoría de nosotros deja la programación a ingenieros porque pensamos que programar y la electrónica son complicadas y difíciles de entender. En la actualidad estas actividades pueden ser divertidas y excitantes. Gracias a Arduino, diseñadores, artistas, personas con un hobby y estudiantes de todas las edades están aprendiendo a crear cosas que se iluminan, mueven y responden ante personas, animales, plantas y el resto del mundo.

Durante años Arduino ha sido usado como "cerebro" en miles de proyectos, cada uno más creativo que el anterior. Una comunidad de colaboradores a nivel mundial ha participado en esta plataforma de código abierto desde personas dedicadas a la programación, fabricantes, los cuales hán contribuido a crear un nuevo mundo de participación, cooperación y colaboración.

Arduino es abierto y simple. Se basa en lecciones que hemos aprendido enseñando en nuestras clases. Si al comenzar asume la idea de que aprender a usar la tecnología digital es simple y accesible, podrá hacerlo. De esta forma la electrónica y la programación podrán ser herramientas que cualquiera pueda usar, como los pinceles y la pintura. Este libro le guía a través de lo básico de una forma sencilla, con proyectos creativos para que los construya y aprenda. Una vez que domine lo básico, tendrá en sus manos programas y circuitos que puede utilizar para crear algo hermoso y hacer que alguien sonría con su invento.

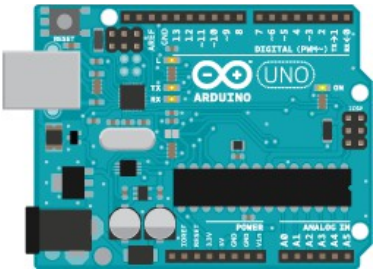
BIENVENIDO A ARDUINO !

Arduino hace los más fácil posible
programar pequeños ordenadores llamados
microcontroladores, los cuales hacen que
que los objetos se conviertan en interactivos

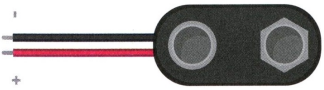
Usted esta rodeado por docenas de ellos cada día: se encuentran dentro de relojes, termostatos, juguetes, mandos de control remoto, hornos de microondas, en algunos cepillos de dientes. Solo hacen una tarea en concreto y si no se da cuenta de que existen – lo cual es en la mayoría de las ocasiones – es porque están realizando bien su trabajo. Han sido programados para detectar y controlar una actividad mediante sensores y actuadores.

Los sensores "escuchan" el mundo físico. Convierten la energía que usted usa al presionar un botón, o al mover los brazos, o al gritar, en señales eléctricas. Botones y mandos son sensores que toca con sus dedos, pero existen otra clase de sensores.
Los actuadores hacen algo dentro del mundo físico. Convierten la energía eléctrica en energía física, como la luz el calor y el movimiento.
Los microcontralodores "escuchan" a los sensores y "hablan" con los actuadores. Ellos deciden que hacer basándose en el programa que tu escribes.
Sin embargo, los microcontroladores y la electrónica que se une con ellos son solo el esqueleto de sus proyectos, tendrá que tener una serie de habilidades para poner algo de carne en los huesos de este esqueleto.
Por ejemplo, en uno de los proyectos que sugerimos, tendrá que hacer una flecha y unirla a un motor para ponerlos juntos en una caja con un mando, de esta forma puede hacer un indicador para decirle a la gente si está ocupado o no lo está. En otro proyecto pondrá algunas luces y un interruptor de inclinación sobre un marco de cartón para hacer un reloj de arena.
Arduino puede hacer que tus proyectos realicen su cometido pero solamente tu puedes hacerlos hermosos. Nosotros te proporcionamos sugerencias en este libro de como conseguirlo.
Arduino fue diseñado para ayudarte a hacer cosas. Para conseguirlo, intentamos que tanto la programación como los materiales electrónicos utilizados se reduzca lo mas posible. Si decides que quieres saber más acerca de estos aspectos, existen muchas buenas guías disponibles. Te proporcionaremos un par de referencias y puedes encontrar más información a través de la página web de Arduino en: arduino.cc/starterkit

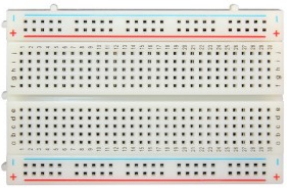
COMPONENTES DE TU KIT



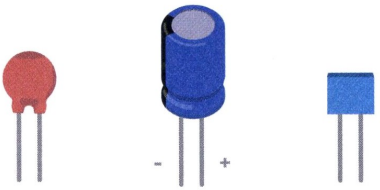
Arduino Uno – La tarjeta de desarrollo del microcontrolador la cual será el corazón de tus proyectos. Es un simple ordenador, pero uno con el cual todavía no puedes realizar nada. Construirás circuitos e interfaces para hacer cosas y decirle al microcontrolador como trabajar con otros componentes.



Clip para Batería – Se utiliza para conectar una batería de 9V a los cables de alimentación y así conectarla fácilmente a la placa Arduino.



Placa de pruebas – Una placa sobre la cual puede montar componentes electrónicos. Es como un panel con agujeros, con filas de agujeros que le permite conectar juntos cables y componentes electrónicos. También están disponibles tarjetas sobre las que hay que soldar y también sin necesidad de usar un soldador como la mostrada aquí.



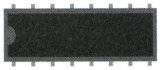
Condensadores – Estos componentes almacenan y devuelven energía eléctrica en un circuito. Cuando el voltaje del circuito es más alto que el que está almacenado en el condensador, la corriente fluye del circuito al condensador, dándole una carga. Cuando la tensión del circuito es mas baja, la energía eléctrica almacenada en el condensador es devuelta al circuito. A menudo se colocan entre los terminales positivo y negativo de una alimentación de un sensor o un motor para ayudar a suavizar las fluctuaciones de tensión que se puedan producir.

Motor de continua (DC) – Convierte la energía eléctrica en energía mecánica cuando la electricidad es aplicada a sus terminales. Una bobina de hilo dentro del motor produce un campo magnético cuando la corriente eléctrica continua (DC) fluye a través de él.

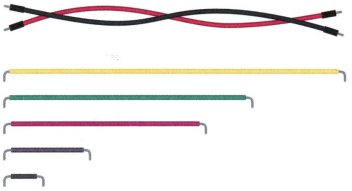
Este campo magnético producido en la bobina atrae y repele al campo magnético de los imanes interiores haciendo que la bobina de hilo gire en el interior. Si se invierte la tensión aplicada el motor gira en sentido contrario.

Diodo – Conduce la electricidad en una sola dirección. Es útil usarlo en un circuito con un motor o una carga que consuma una gran cantidad de corriente eléctrica. Los diodos tienen polaridad, esto quiere decir que hay que colocarlos de una forma determinada (polarizado) dentro del circuito. Colocado de esta manera (correctamente polarizado) permite que la corriente eléctrica pase a través de él. Colocado al revés (inversamente polarizado) no deja pasar la corriente eléctrica. El diodo tiene dos terminales, uno de ellos llamado ánodo, el cual se conecta dentro de un circuito al punto donde más tensión existe. El otro terminal llamado cátodo, se conecta a otro punto con una tensión inferior con respecto al punto en donde se conecta el ánodo. El cátodo normalmente se indica mediante una franja de color blanco en uno de los lados del cuerpo del diodo.

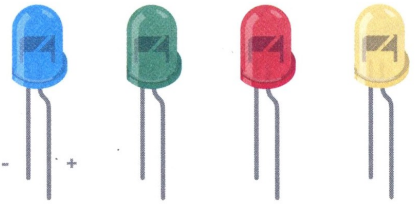
Papel celofán (rojo, verde, azul) – Son filtros que dejan pasar diferentes longitudes de onda de la luz. Cuando se utilizan con una foto resistencia (sensor), pueden hacer que el sensor solo reaccione a una cantidad de luz que entra en el filtro de color, de manera que otras luces de otras longitudes de onda no hacen reaccionar al sensor.



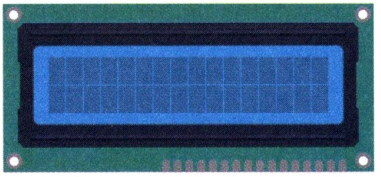
Puente-H – Se trata de un circuito que permite controlar la polaridad de la tensión aplicada a un carga. El puente-H en el kit es un circuito integrado, pero se puede construir a partir de un número determinado de componentes discretos (resistencias, condensadores y transistores).



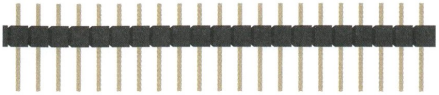
Cables puente – Utilizarlos para conectar unos componentes con otros sobre la placa de prueba, y la tarjeta de Arduino.



Diodos Emisores de Luz (LEDs) – Un tipo de diodo que emite luz cuando la corriente lo atraviesa. Como en todos los diodos, la corriente solo fluye en un sentido a través de estos componentes. Estará probablemente familiarizado con ellos al verlos como indicadores dentro de una gran variedad de dispositivos electrónicos. El ánodo, que normalmente se conecta al positivo de la alimentación, es generalmente el terminal mas largo, y el cátodo el terminal mas corto.



Pantalla de Cristal Líquido (LCD) – Un tipo de pantalla numérica o gráfica basado en cristal líquido. Los LCDs están disponibles en varios tamaños, formas y estilos. El que se incluye con este kit dispone de 2 filas con 16 caracteres en cada una de ellas.



Tira de pines macho – Estos pines se conectan en zócalos hembra, como los que tiene una placa de pruebas. Permiten conectar otros elementos electrónicos con mucha facilidad.

Optoacoplador – Permite conectar dos circuitos que no tienen en común la misma fuente de alimentación. En su interior hay un pequeño diodo led que, cuando se ilumina, hace que un foto-receptor cierre un interruptor interno. Cuando se aplica una tensión al terminal + (positivo), el diodo led emite luz y el interruptor interno se cierra. Las dos salidas reemplazan a un interruptor en el circuito secundario.

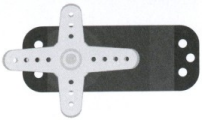
Zumbador piezo eléctrico – Un componente eléctrico que se puede usar para detectar vibraciones y generar ruidos.

Foto resistencia – (también llamada foto célula o resistencia dependiente de la luz). Se trata de una resistencia variable que cambia su resistencia según el nivel de luz que incide sobre su superficie.

Potenciómetro – Una resistencia variable con tres terminales. Dos de estos terminales están conectados a los extremos de una resistencia fija. El terminal central se puede mover a través de la superficie de la resistencia fija (dispone de un mando), consiguiendo de esta forma dos valores diferentes de resistencia según el terminal extremo que se tome como referencia. Cuando los terminales extremos del potenciómetro se conectan entre una tensión y masa, en el terminal central aparece una tensión que es proporcional al giro del mando central, entre cero (un extremo) y la máxima tensión (el otro extremo).

Pulsador – Interruptores momentáneos que cierran un circuito cuando son presionados. Se colocan con facilidad sobre la placa de pruebas. Son buenos para abrir o cerrar el paso a una señal.

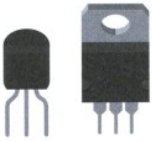
Resistencias – Se opone al paso de la corriente eléctrica en un circuito, dando como resultado a un cambio en la tensión y en dicha corriente. El valor de las resistencias se mide en ohmios (se representa por la letra griega omega: Ω). Las bandas de colores en un lado de la resistencia indica su valor (ver la tabla de código de colores de la resistencia en la página 41).



Servo motor – Un tipo de motor reductor que solo puede girar 180 grados. Es controlado por las señales eléctricas en formato de pulsos que son enviadas desde la tarjeta Arduino. Estos pulsos le dicen al motor a que posición se debe de mover.

Sensor de temperatura – Cambia la tensión de salida que suministra dependiendo de la temperatura que tenga su encapsulado. Sus terminales extremos se conectan entre una tensión y masa. El voltaje del terminal central cambia según este componente este más caliente o más frío.

Sensor de inclinación – Un tipo de interruptor que se abre o se cierra dependiendo de su orientación. Normalmente son cilindros huecos con una bola de metal en su interior la cual hará que los dos terminales se unan a través de esta bola cuando se incline en una determinada dirección.



Transistor – Componente de tres terminales que puede trabajar como un interruptor electrónico. Es útil para controlar corrientes y tensiones grandes como la de los motores. Un terminal se conecta a masa, otro terminal a un elemento que se quiera controlar (motor, bombilla, zumbador) y el tercer terminal se conecta a una salida de Arduino. Cuando el transistor recibe una tensión de control a través del terminal que está conectado a Arduino, cierra los terminales extremos, entre masa y el terminal donde se conecta el elemento, de manera que dicho elemento recibe la energía necesaria que lo hace funcionar (gira, emite luz, genera un sonido).

Cable USB – Permite conectar la placa Arduino Uno a un ordenador para que se pueda programar. También proporciona la alimentación necesaria tanto a la placa Arduino como a todos los componentes electrónicos que forman parte de los proyectos de este kit.

TABLA CON LOS SÍMBOLOS DE LOS COMPONENTES ELECTRÓNICOS

Cables conectados

Cables no conectados

Sensor de inclinación

Resistencia

Diodo Led

Condensador polarizado

Tierra

Transistor Bipolar

Transistor Mosfet

Resistencia LDR

Diodo

Condensador

Pulsador

Motor

Potenciómetro

Zumbador piezo-eléctrico

Batería

En este libro le mostraremos los circuitos de dos formas distintas, como ilustraciones realistas y como esquemas electrónicos. Las ilustraciones le dan una idea de como podrán quedar montados los componentes electrónicos en la placa de pruebas para realizar un proyecto. Los esquemas, en su lugar, utilizan símbolos para presentar en esencia como funciona el circuito: son una forma de representar la conexión entre los componentes de una forma clara y concisa pero no muestran como se van a montar sobre la placa de pruebas. Los esquemas y los símbolos son la forma en como se representan los circuitos electrónicos. A medida que explore el mundo de la electrónica descubrirá que algunos libros y páginas web solo muestran los esquema de los circuitos electrónicos, así que aprender a entender como funciona un circuito de esta forma es una habilidad muy valorada. En esta hoja se muestran los símbolos de los componentes electrónicos que se usarán en este libro.

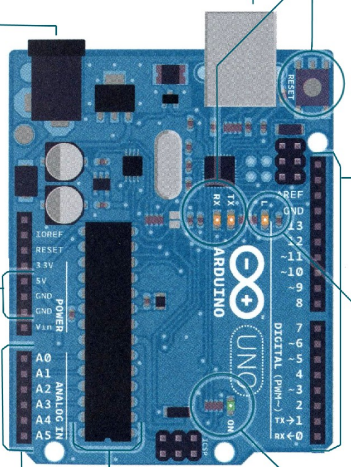
LA PLACA ARDUINO

Conector de alimentación
Este conector se utiliza para alimentar la placa Arduino cuando no esta conectada a un puerto USB.
Acepta tensiones entre 7 y 12V

Puerto USB
Usado para alimentar y cargar los programas a su Arduino, y para la comunicación con el programa de Arduino (mediante la instrucción Serial.println() etc.)

Botón de reset
Puesta a cero del micro-controlador ATmega

LEDs TX y RX
Estos diodos LEDs indican cuando se realiza una comunicación entre Arduino y el ordenador. Parpadean rápidamente cuando se carga el programa así como durante la comunicación serie. Útil para la depuración.



Pins Digitales
Usar estos pins con las instrucciones digitalRead(), digitalWrite(), y analogWrite(). La instrucción analogWrite() solo trabaja con los pins con el símbolo PWM

Pin 13 LED
El único componente que actúa como dispositivo de salida incorporado a su Arduino Uno. Lo usará cuando ejecute su primer programa. Este LED es muy útil para la depuración.

Pines GND y 5V
Usar estos pins para proporcionar una tensión de +5V y masa para los circuitos externos a la placa.

Microcontrolador ATmega
El corazón de la placa Arduino Uno

Led de Encendido
Indica que la placa Arduino está siendo alimentada. Útil para la depuración.

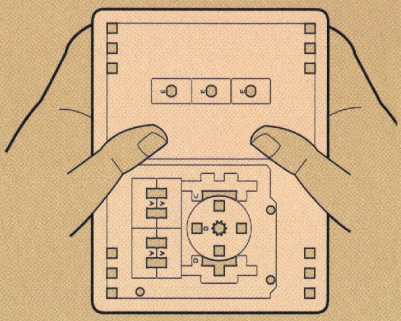
Entradas Analógicas
Usar estos pins con la instrucción analogRead()

Su Kit de Inicio incluye una base de madera pre-cortada de ensamblaje fácil para montar en ella todos sus proyectos, tanto los que son de este libro como aquellos que no lo son, será más sencillo.

Para montarla, sacar la base de madera de la caja y seguir las instrucciones que se muestran en las imágenes de la derecha.

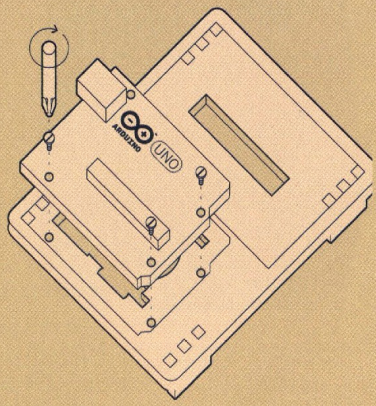
Tenga cuidado de utilizar sólo las piezas que se muestran, y no perder ninguna de ellas: las cuales necesitará para algunos de los proyectos que se incluyen en esta obra.

¡Así que adelante!



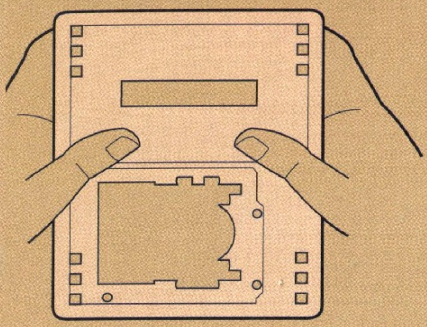
1

Separar con cuidado en piezas el tablero de madera



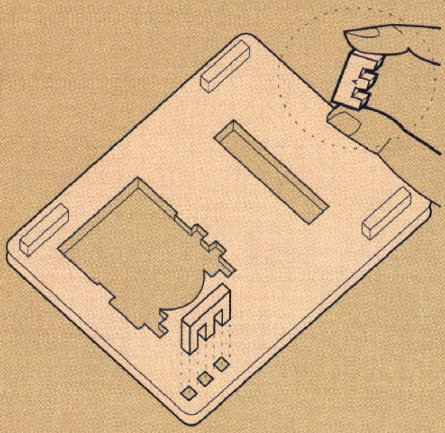
4

Sujetar la placa Arduino Uno a la base de madera usando tres tornillos, Tener cuidado al apretarlos.



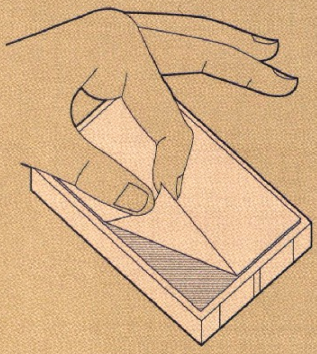
2

Presionar hasta separar todas las piezas



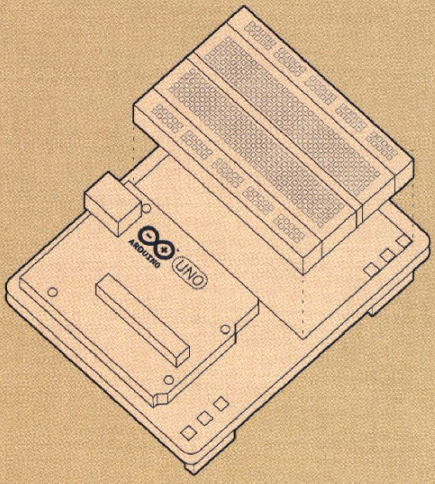
3

Colocar las piezas marcadas con una "A" dentro de los agujeros de las esquinas, con la finalidad de crear el soporte para la base



5

Retire con cuidado el papel protector de la placa de pruebas



6

Pegar la placa de pruebas sobre la base de madera, próxima a la Arduino Uno

OTRAS COSAS QUE VA A NECESITAR

Batería de 9 voltios

Una pequeña fuente de luz como una linterna

Material conductor como
papel de aluminio o malla de cobre

Papel de color

Tijeras

Un viejo CD o DVD

Cinta y goma

Una caja a la que le ha perforado
unos agujeros

Herramientas básicas como un destornillador

Un conector con cable para la batería de 9V
Cualquier conector con cable que tenga por lo menos un interruptor o un pulsador, y que este dispuesto a usar, valdrá pare realizar este trabajo

Soldador de estaño y estaño
(solo es necesario en el Proyecto 15)

LA PREPARACIÓN

ANTES DE QUE COMIENCE A CONTROLAR EL MUNDO A SU ALREDEDOR, NECESITA DESCARGAR EL ENTORNO INTEGRADO DE DESARROLLO (IDE) DE ARDUINO PARA PODER PROGRAMARLO

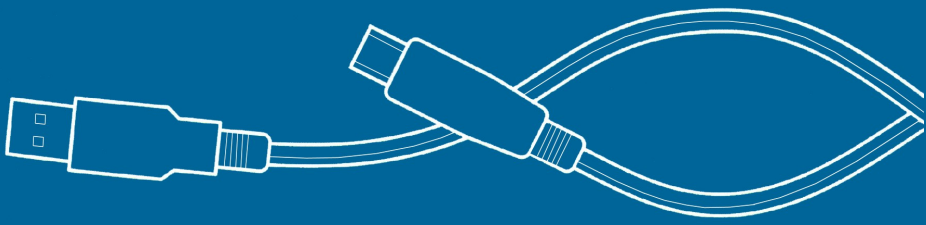
El IDE de Arduino le permite escribir los programas y cargarlos a su placa Arduino.

Descargar la última versión del IDE desde:
arduino.cc/download

Coloque la placa Arduino y el cable USB cerca de su ordenador. Todavía no los conecte.

Siga las instrucciones que se muestran en las siguientes páginas para realizar la instalación para Windows, Mac OS X o Linux.

La versión online de esta guía esta disponible en la siguiente dirección:
arduino.cc/guide



INSTALACIÓN EN WINDOWS

Versión online

arduino.cc/windows

WINDOWS 7, VISTA, Y XP

1. Después de finalizar la descarga hacer doble click sobre el fichero “Install Arduino”. Si se abre una ventana emergente de seguridad de Windows, pulsar sobre “Ejecutar” o “Permitir” y aceptar los términos de la licencia al pulsar sobre el botón “I Agree”. Pulsar sobre “Next” para seleccionar la carpeta en donde se va a instalar el IDE y entonces pulsar sobre el botón “Install”.
2. Después de que el software de Arduino se haya instalado, conectar la placa Arduino al ordenador a través del cable USB. La placa se alimenta automáticamente a través de esta conexión USB del ordenador, en ese momento el LED verde (etiquetado como ON) en la placa Arduino se encenderá.
3. Windows inicia el proceso de instalación del controlador cuando la placa se conecta. El ordenador no podrá encontrar el controlador sin ayuda, es necesario indicarle en que carpeta se localiza.
   a) **Windows XP:** Si el programa de actualización de Windows pregunta acerca de la ruta del controlador, seleccionar “Sí, sólo esta vez” y después “Instalar desde una lista o una ubicación específica (recomendado)”;
   b) **Vista o Windows 7:** En Windows 7 si se abre una ventana emergente en donde le pide si instalar el controlador automáticamente o buscarlo en el ordenador escoger buscar el controlador en el ordenador. En Vista, continuar con el paso siguiente al escoger la opción recomendada.
4. Si la instalación no comienza automáticamente, pulsar sobre el botón de Inicio y abrir el Panel de Control. Entonces, dirigirse al Administrador de Dispositivos siguiendo estos pasos:
   a) **Windows XP:** Cambiar a Vista Clásica → Systema → Hardware → Administrador de Dispositivos
   b) **Windows Vista:** Vista Clásica → Administrador de Dispositivos
   c) **Windows 7:** Sistema y Seguridad → Sistema → Administrador de Dispositivos
   d) Buscar el dispositivo Arduino dentro de la categoría “Otros Dispositivos” o “Dispositivos Desconocidos” y seleccionar “Actualizar Controlador” o “Actualizar Controlador del programa” al pulsar el botón derecho del ratón.
5. Pulsar sobre “Examinar” y seleccionar la carpeta “Controladores” (no la carpeta de los controladores USB FTDI) de la carpeta de Arduino. Presionar “OK” y “Siguiente”. Si se abre una ventana emergente con el logo de Windows, pulsar sobre “Continuar de todas formas”. Ahora Windows instala el controlador.
6. Dentro de la ventana de Administración de Dispositivos, bajo la categoría de “Puertos (COM & LPT)” podrá ver un puerto similar a “Arduino UNO (COM4)”.

¡Enhorabuena! Acaba de instalar el IDE de Arduino en su computadora.

INSTALACIÓN EN MAC OS X

OS X 10.5 Y POSTERIORES

Versión online

arduino.cc/windows

1. Si usted está usando la versión 10.8 (Mountain Lion) o posterior, dirigirse a las "Preferencias del Sistema" y abrir el panel de "Seguridad & Privacidad". En la pestaña "General" bajo el encabezado "Permitir descargar aplicaciones desde", hacer click en "Desde cualquier lugar".
2. Una vez que el IDE de Arduino se ha descargado, hacer doble click sobre el fichero zip para descomprimirlo.
3. Copiar la aplicación Arduino a la carpeta de Aplicaciones, o cualquier otro sitio en donde desee instalar el software.
4. Conectar la placa Arduino al ordenador a través del cable USB. La placa se alimenta automáticamente a través de esta conexión USB del ordenador, en ese momento el LED verde (etiquetado como ON) en la placa Arduino se encenderá.
5. No necesita instalar ningún controlador para trabajar con la placa.
6. Dependiendo de la versión del sistema operativo OS X que se este ejecutando, podría aparecer una ventana emergente con un mensaje pidiéndole si desea abrir las "Preferencias del Sistema". Hacer click sobre el botón de "Preferencias de Red" y después en "Aplicar".
7. El gestor de ventanas Uno de Mac mostrará "No configurado", pero el entorno IDE ya está listo. Pude salir de "Preferencias del Sistema"

¡Enhorabuena! El entorno de Arduino ha sido instalado y ya está listo para comenzar con los proyectos

INSTALACIÓN EN LÍNUX

Si usted está usando Linux, por favor visite la página web para ver como se hace.

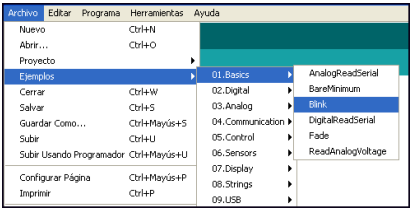
Arduino.cc/linux

COMUNICACIÓN CON ARDUINO

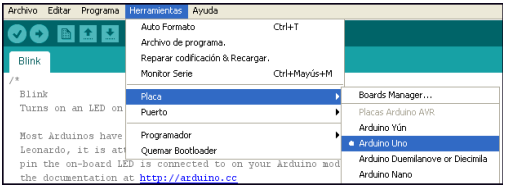
Ahora que ya ha instalado el IDE de Arduino y esta seguro de que su ordenador puede comunicarse con la placa, es el momento de comprobar si puede cargar un programa en Arduino

1. Hacer doble click sobre la aplicación de Arduino para abrirla. Si el entorno IDE esta en un lenguaje que no es el suyo, puede cambiarlo al seleccionar el menú “Archivo” y escoger “Preferencias”. En la ventana que se abre y dentro del “Editor de idioma” escoger el lenguaje que desee. Reiniciar el programa para que tenga efecto. Buscar “the environment page ” dentro de esta página para obtener más información: arduino.cc/ide

2. Navegar para abrir un sketch de ejemplo que hace que el diodo led de la placa Arduino parpadee (la palabra ´sketch´es como se llama a los programas en Arduino).
Seleccionar el menú “Archivo” escoger “Ejemplos” a continuación “01Basics” y por último “Blink”.



3. Debería de abrirse una nueva ventana con un texto dentro. Dejar la ventana como esta por ahora, y seleccionar su placa Arduino desde:
Herramientas > Placa



4. Escoger el puerto serie en donde su placa Arduino esta conectada desde el menú de:
Herramientas > Puerto

   a) **En Windows:** Probablemente aparecerá con el número más alto de puerto COM. No pasará nada si se equivoca al escoger este número de puerto, simplemente no funcionará.

   b) **En Mac:** Deberá aparecer algo parecido a /dev/tty.usbmoderm en este sistema. Por lo general aparecen dos, seleccionar uno cualquiera de ellos.

Fig. 1

5. Para cargar el Sketch que hace que el diodo led parpadee a su Arduino, presionar el botón ***Subir*** en la esquina superior izquierda de la ventana. Ver la figura 1

6. Debe de ver una barra indicando el progreso de carga del sketch cerca de la esquina inferior derecha del IDE de Arduino, y los diodos led de la placa Arduino con las etiquetas TX y RX estarán parpadeando en el momento de la carga. Si la carga se ha realizado correctamente, el IDE mostrará el mensaje ***SUBIDO*** en la esquina inferior izquierda.

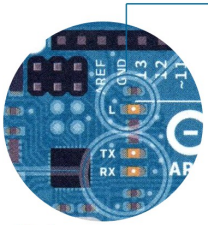


Fig. 2

7. A los pocos segundos de completar la carga del Sketh, debe de ver como el diodo led amarillo, con la etiqueta L cerca, comienza a parpadear. Ver figura 2.
Si es así, ¡felicidades! Acaba de conseguir programar Arduino para que el diodo led de la placa parpadee.

   A veces su nuevo Arduino está ya programado con el Sketch de parpadeo, así que no puede saber si realmente lo acaba de programar. En este caso, cambiar dentro de la instrucción "delay" el tiempo que aparece entre paréntesis a 100, y volver a subir de nuevo el sketch de parpadeo. Ahora el diodo led de la placa debe de parpadear más rápido.

   ¡Felicidades! ¡Usted realmente tiene el control!. Ahora es el momento de comenzar con el proyecto número 1. (No necesita guardar ninguno de los cambios que ha realizado.)

INFORMACIÓN ADICIONAL

Si tiene problemas con alguno de los pasos descritos anteriormente, por favor mirar las sugerencias para solucionar los problemas a través de la siguiente página web:
arduino.cc/trouble

Mientras se esta preparando para realizar los proyectos de este libro, puede dirigirse a la siguiente página para obtener más información acerca del entorno de programación de Arduino.
arduino.cc/en/Guide/Environment

También puede visitar estas dos páginas:

- Ejemplos de utilización de sensores y actuadores
arduino.cc/tutorial

- La referencia para el lenguaje de Arduino
arduino.cc/examples

01

PULSADOR

DIODO LED

RESISTENCIA DE 220 OHMIOS

INGREDIENTES

CONOZCA SUS HERRAMIENTAS

USTED VA A MONTAR UN CIRCUITO SENCILLO CON ALGUNOS PULSADORES, UN DIODO LED, Y UNA RESISTENCIA

Descubra: La teoría básica de la electricidad, como se trabaja con una placa de prueba y como se conectan componentes en serie y en paralelo

Tiempo: **30 MINUTOS**

Nivel: **bajo**

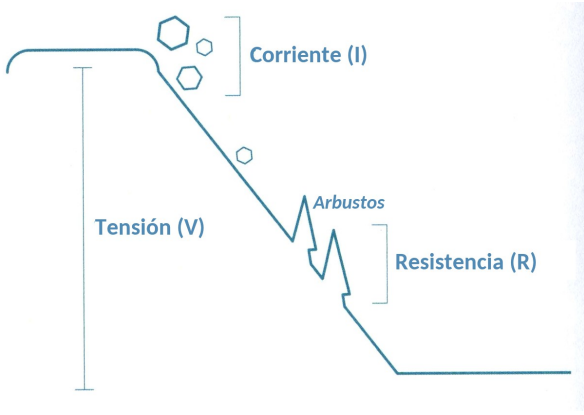
La electricidad es un tipo de energía, como el calor, la gravedad o la luz. La energía eléctrica fluye través de los conductores, como los cables. Puede transformar la energía eléctrica en otras formas de energía para hacer algo interesante, como encender una luz o hacer algo de ruido a través de un altavoz.

Los componentes que podría usar para hacer esto, como altavoces o bombillas, son **transductores** eléctricos. Los transductores transforman otros tipos de energía en energía eléctrica y viceversa. Los componentes que transforman otras formas de energía en energía eléctrica son a menudo llamados **sensores**, y los componentes que convierten la energía eléctrica en otras formas de energía se conocen algunas veces con el nombre de **actuadores**. Construirá **circuitos** para hacer que la electricidad circule a través de diferentes componentes. Se trata de circuitos en bucles cerrados mediante cables con una fuente de energía (como una batería) y algún componente que haga algo útil con la energía; este componente se suele llamar carga.

En un circuito, la electricidad fluye desde un punto con el potencial de energía más alto (normalmente se conoce como el positivo o + de la fuente de energía) a un punto con el potencial de energía más bajo. La masa (a menudo representado con el signo "-" o GND) es normalmente el punto con el menor potencial de energía en un circuito. En los circuitos que va a construir la corriente eléctrica solo circula en una dirección. Este tipo de circuitos se llaman de corriente directa o DC. Por otro lado en los circuitos con corriente alterna (AC) la electricidad cambia de dirección 50 o 60 veces por segundo (dependiendo de donde usted viva). Este es el tipo de electricidad que puede encontrar en un enchufe de la pared de una habitación.

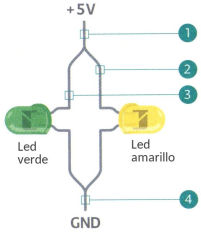
Existen una serie de términos con los cuales debe de familiarizarse cuando trabaje con circuitos eléctricos. **Corriente** (medida en amperios, o amps; con el símbolo **A)** es la cantidad de carga eléctrica que circula a través de un determinado punto de un circuito. **Tensión** (medido en voltios; con el símbolo **V)** es la diferencia de energía entre un punto de un circuito y otro que se toma como referencia. Y por último, **resistencia** (medida en ohmios; con el símbolo **Ω**) representa cuanto se opone un componente a que la energía eléctrica fluya a través de él.

Una forma de imaginar como funciona todo lo explicado con anterioridad es imaginar un acantilado con una pendiente por donde se deslizan unas rocas cuesta a bajo, como se muestra en la figura 1. Cuando mayor sea el acantilado mayor energía tendrán las rocas para llegar hasta la parte inferior. La altura del acantilado es como la tensión en un circuito: cuando mayor es el voltaje en la fuente de energía, mayor energía puede usar. Cuanto más rocas tenga, mas cantidad de energía se desplaza hacía abajo por el acantilado. El número de rocas es como la corriente en un circuito eléctrico. Las rocas al desplazarse por la pendiente del acantilado tropiezan con los arbustos y pierden algo de energía al aplastarlos y poder pasar por encima de ellos. Los arbustos son como las resistencias en un circuito, ofreciendo una oposición al paso de la electricidad a la vez que la convierten en otras formas de energía (como calor, sonido, etc).



Desplazamiento de las rocas para entender como fluye la corriente eléctrica
Figura 1

UN PAR DE COSAS SOBRE LOS CIRCUITOS



La corriente en (1) = corriente en (2) + corriente en (3) = corriente en (4)

Figura 2

- En un circuito es necesario que exista un camino desde la fuente de energía (alimentación) hasta el punto de menor energía (masa). Si no existe un camino por donde la energía se pueda mover, el circuito no funcionará.

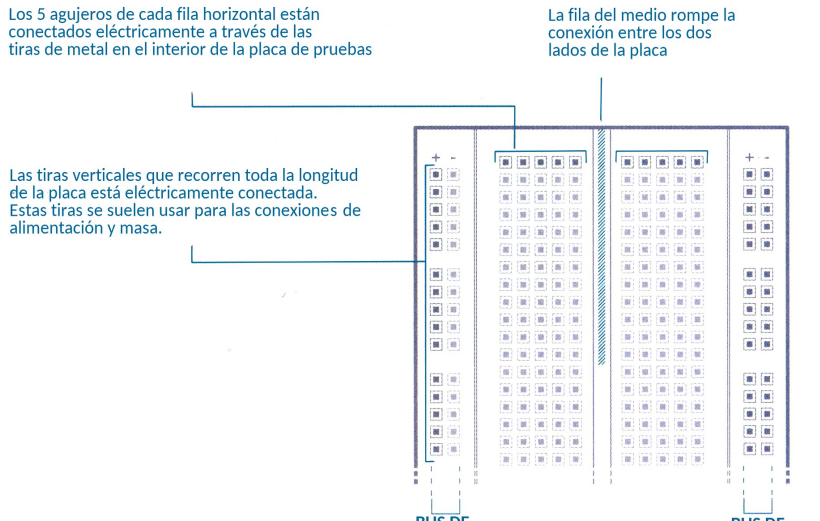
- Toda la energía eléctrica es utilizada por los componentes que forman parte de un circuito. Cada componente convierte parte de esa energía en otra forma de energía. En cualquier circuito, todas las tensiones se convierten en otra forma de energía (luz, calor, sonido, etc.).

- El flujo de corriente en un punto específico en un circuito siempre será el mismo que entra y que sale.

- La corriente eléctrica siempre busca el camino de menor resistencia hacia masa. Si existen dos caminos posibles, la mayoría de la corriente eléctrica circulará por el camino con menor resistencia. Si dispone de una conexión en donde se conectan los puntos de alimentación y masa juntos directamente y sin resistencia, se producirá un cortocircuito; la corriente será demasiado grande al no disponer de una resistencia que reduzca su valor. En un cortocircuito, la fuente de alimentación y los cables convierten la energía eléctrica en luz y calor, se producirán chispas y/o una explosión. Si alguna vez ha cortocircuitado una batería y a visto chispas sabrá lo peligroso que un cortocircuito que puede ser.

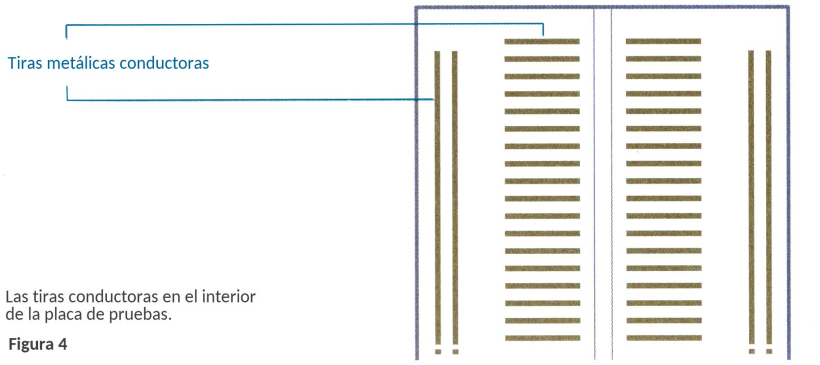
QUÉ ES UNA PLACA DE PRUEBAS

La placa de pruebas es el primer lugar en donde montará sus circuitos. La que se incluye en el kit no necesita soldar nada para montar los componentes encima, es como un juego de LEGO en formato electrónico. Las filas verticales y horizontales de la placa de pruebas, como se muestra en la figura 3, conducen la electricidad a través de los conectores de metal fino que hay debajo del plástico con agujeros.



La parte superior de la placa de pruebas y las conexiones que hay debajo

Figura 3



Las tiras conductoras en el interior de la placa de pruebas.

Figura 4

DIBUJOS DE UN CIRCUITO

A través de estos proyectos, podrá ver dos vistas de los circuitos: una vista sobre la placa de pruebas (como en la figura 5), que se parece a las cosas de su kit. La otra es la vista de esquema (como en la figura 6), se trata de una forma más abstracta de mostrar las relaciones entre los componentes de un circuito. Un esquema no siempre muestra donde se colocan los componentes unos con respectos a otros, pero sí que muestra como están conectados.

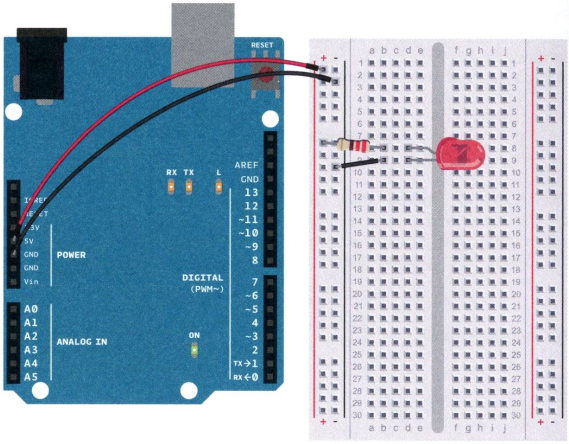
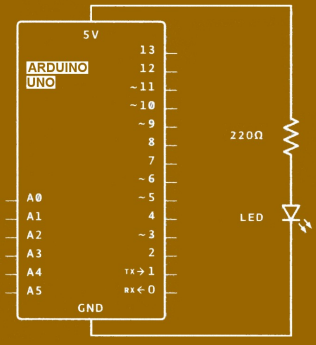


Ilustración del circuito

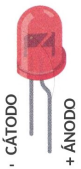
Figura 5



Vista de esquema

Figura 6

SUS PRIMEROS COMPONENTES

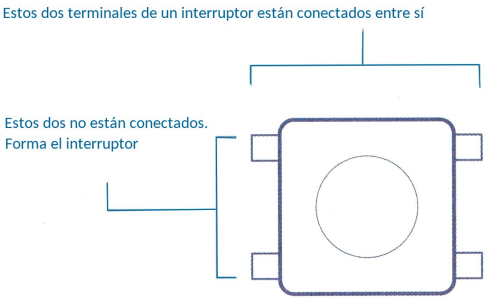


Un ***LED***, o diodo emisor de luz , es un componente que convierte la energía eléctrica en energía luminosa. Los LEDs son componentes que tienen polaridad, esto quiere decir que solo circula corriente a través de ellos en una sola dirección. El terminal más largo del LED es llamado ánodo, se conectará a la alimentación. El terminal más corto es el cátodo y se conectará a masa. Cuando la tensión es aplicada al ánodo del led y el cátodo está conectado a masa, el LED emite luz.

Una ***resistencia*** es un componente que se opone al paso de la energía eléctrica (ver el listado de componentes para obtener información sobre las bandas de colores que tiene en un lado del cuerpo de la resistencia). Transforma parte de la energía eléctrica en calor. Si se coloca una resistencia en serie con un componente como un LED, el resultado será que el diodo led recibe menos energía al consumir la resistencia esa energía que el LED no recibe. Esto permite poder alimentar a los componentes con la cantidad de energía que necesitan. Puede usar una resistencia en serie con un LED para evitar que reciba demasiada tensión. Sin la resistencia, el led podría brillar con gran intensidad durante unos momentos, pero rápidamente se quemará.

Un ***interruptor*** interrumpe la circulación de la electricidad, abriendo el circuito cuando se abre. Cuando un interruptor está cerrado, permite que el circuito se alimente. Hay muchos tipos de interruptores. Algunos de los que se incluye en el kit se llaman interruptores momentáneos, o pulsadores, porque solo se cierran cuando son presionados.

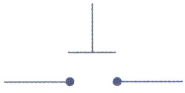
CONEXIONES DEL INTERRUPTOR



El interruptor
Figura 7

SÍMBOLOS EN EL ESQUEMA

A - Símbolo como interruptor



B - Símbolo como pulsador

MONTANDO EL CIRCUITO

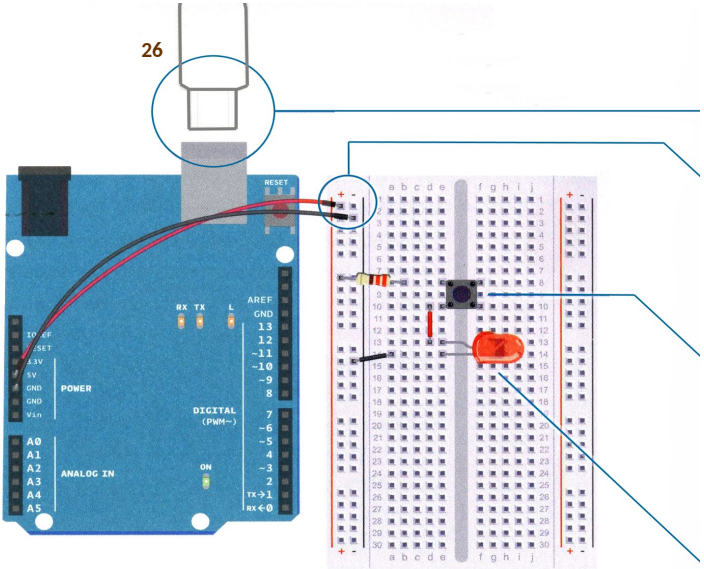


Figura 8

> *Su primer circuito interactivo, usando un pulsador, una resistencia y un LED. Arduino solo se utiliza como fuente de alimentación para este circuito. En posteriores proyectos, conectará sus terminales de entrada y de salida para controlar circuitos más complejos*

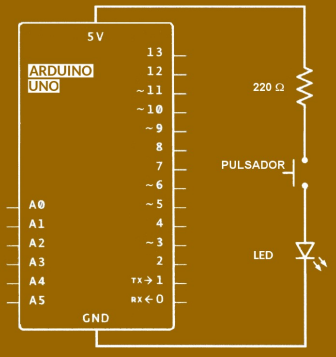


Figura 9

Va a usar la placa Arduino en este proyecto, pero solo como una fuente de alimentación. Cuando la conecte a un puerto USB o a una batería de 9V, Arduino suministrará una tensión de 5V entre su terminal de 5V y su terminal de masa los cuales puede usar. 5V = 5 voltios, lo verá escrito de esta forma muchas veces.

1. Si la placa Arduino está conectada a una batería o a un ordenador vía USB, ¡desconectarla antes de montar el circuito!

2. Conectar un cable rojo al terminal de 5V de Arduino y conectar el otro extremo de este cable en una de las columnas del bus de la placa de pruebas marcada con el símbolo +. Conectar el terminal de masa de Arduino con un cable negro a la línea adyacente en donde se ha conectado el cable rojo y en la columna marcada con el símbolo -. Es útil para guardar una relación entre los colores de los cables (rojo para la alimentación y negro para la masa) a lo largo del circuito.

3. Ahora que ha alimentado la placa, coloque el pulsador en el centro de la placa de pruebas. El pulsador se situá en el centro en una dirección. La parte curva de los terminales del pulsador apuntan hacia el centro de la placa.

4. Usar una resistencia de 220 ohmios para conectar la alimentación (columna marcada con +)a uno de los lados del pulsador. Las ilustraciones de las resistencias en este libro son con 4 bandas. Su kit puede tener una mezcla de resistencias con 4 y 5 bandas. Usar la ilustración adjunta para verificar que está usando la resistencia adecuada en este proyecto. Mirar la página 41 para obtener más información sobre el código de colores de las resistencias.
En el otro lado del pulsador, conectar el ánodo (terminal largo) del diodo LED. Con un cable conectar el cátodo (terminal corto) del LED a masa. Cuando este todo listo, enchufar el cable USB a la placa Arduino.

ÚSALO

Una vez que todo esta preparado, presionar el botón. El diodo LED deberá encenderse. ¡Le felicito, acaba de que conseguir que el circuito funcione! Una vez que se haya cansado de presionar el botón para encender la luz, es el momento de mejorar las cosas añadiendo un segundo botón.

Usted va a colocar componentes sobre la placa de pruebas en serie y en paralelo. Los componentes en serie se colocan unos detrás de otros y los componentes en paralelo se conectan uno al lado del otro

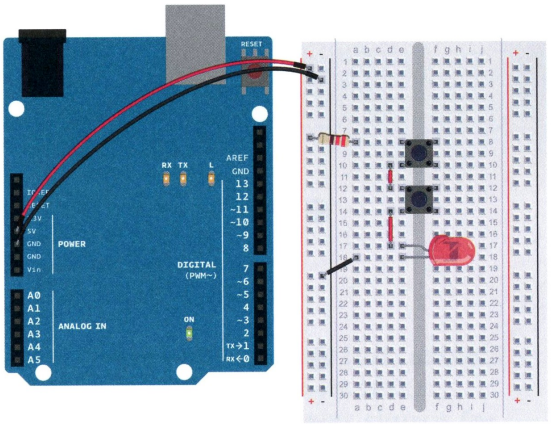
Estos dos elementos están en serie

Circuito en serie

LOS COMPONENTES EN SERIE SE CONECTAN UNO DESPUÉS DE OTRO

Una vez que ha desconectado la fuente de alimentación añadir un pulsador cerca del que ya está montado en la placa de pruebas. Conectar un cable para conectarlos en serie como se muestra en la figura 10. Conectar el ánodo (terminal largo) del LED al segundo pulsador. Conectar el cátodo del LED a masa. Alimentar de nuevo la placa Arduino: ahora para encender el LED, necesita presionar los dos pulsadores a la vez. Puesto que están en serie, ambos deben ser cerrados para que el circuito funcione.

QUITAR LA ALIMENTACIÓN ANTES DE CAMBIAR ALGO EN SU CIRCUITO



Los dos pulsadores están en serie. Esto quiere decir que circula la misma corriente a través de ellos, así que ambos tienen que ser presionados a la vez para que el LED se encienda

Figura 10

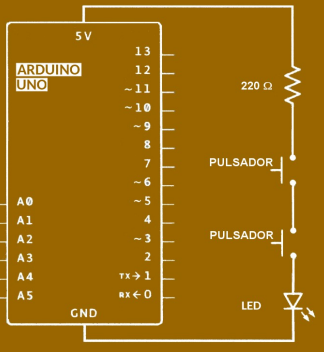
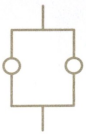


Figura 11

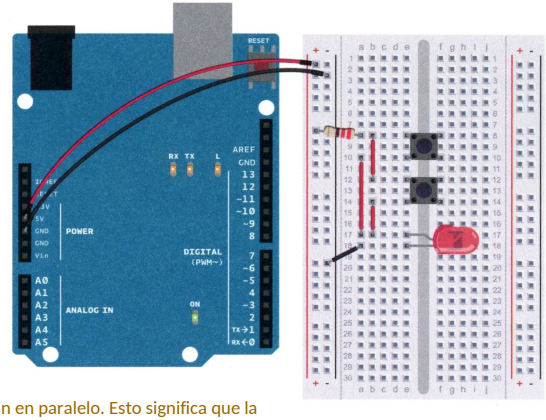


Estos dos elementos están en paralelo

Circuito en paralelo

LOS COMPONENTES EN PARALELO SE CONECTAN UNO AL LADO DEL OTRO

Ahora que ha dominado el arte de las cosas en serie, es el momento de conectar los pulsadores en paralelo. Dejar los pulsadores y el diodo LED donde están, pero quitar la conexión entre los dos pulsadores. Colocar un cable desde cada pulsador a la resistencia. Unir el otro extremo de cada pulsador al diodo LED, como muestra la figura 12. Ahora cuando se presiona cualquier botón, el circuito funciona y el diodo led se enciende.



Estos dos pulsadores están en paralelo. Esto significa que la corriente eléctrica se divide entre ellos. Si cualquier pulsador es presionado, el diodo LED se encenderá.

Figura 12

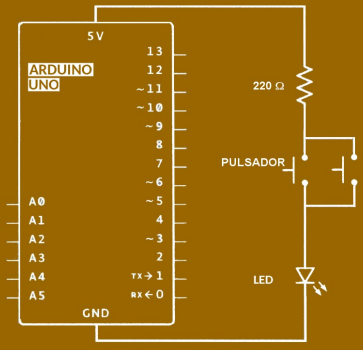
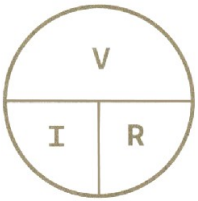


Figura 13

ENTENDIENDO LA LEY DE OHM



Puede usar este círculo para recordar las relaciones entre voltaje, corriente y resistencia. Ponga su dedo sobre cualquiera de los tres, y verá como se relaciona con los otros dos.

V = I * R

I = V / R

R = V / I

Corriente, voltaje y resistencia están todos relacionados. Cuando cambia uno de estos parámetros en un circuito, afecta a los demás. La relación que existe entre ellos se conoce como ley de Ohm, en honor a Georg Simon Ohm quien la descubrió.

TENSIÓN (V) = CORRIENTE (I) x RESISTENCIA

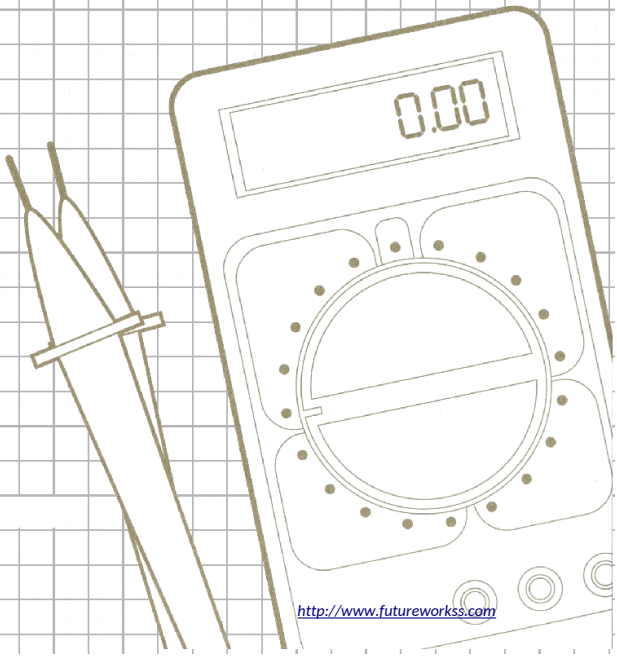
Al medir intensidad (amperios) en los circuitos que vaya a montar, los valores serán del rango de miliamperios. Un miliamperio vale la milésima parte de un amperio.

En el circuito de la figura 5 se aplica una tensión de 5 voltios. La resistencia tiene un valor de 220 ohmios. Para averiguar la corriente que usa el LED, reemplazar los valores en la ecuación de la Ley de Ohm. Por tanto I = V / R ; I = 5 / 220 = 0.023 amperios, este valor equivale a la 23 milésima parte de un amperio, o 23 miliamperios (23mA) que consume el diodo LED. Este valor es casi el máximo con el cual puede trabajar con seguridad este tipo de diodo LED, es por eso que se utiliza un resistencia de 220 ohmios.

Puede usar este proyecto de varias formas diferentes, o creando su propio pulsador (se puede realizar con dos trozos de papel de aluminio con un cable), o creando una combinación de pulsadores y LEDs en paralelo y en serie. ¿Qué sucede cuando coloca tres o cuatro LEDs en serie?, ¿qué sucede cuando se colocan en paralelo?, ¿por qué funcionan de la forma en lo hacen?

Un ***multímetro o polímetro*** es un instrumento de medida con el cual puede medir la cantidad de resistencia, corriente y voltaje en un circuito. Por ahora no es necesario usarlo en estos proyectos, pero es importante disponer de uno para buscar una avería cuando un circuito no funcione. Puede ver una buena descripción de como usar uno a través de esta página web ***manejo básico del polímetro***

Ha aprendido acerca de las propiedades del voltaje, la corriente y la resistencia mientras construía un circuito sobre una placa de pruebas. Con algunos componentes como LEDs, resistencias y pulsadores, puede crear un sistema interactivo muy simple: un usuario presiona un botón y una luz se enciende. Estos conocimientos sobre electrónica se volverán a usar y se ampliarán en próximos proyectos.



02

PULSADOR

DIODO LED

RESISTENCIA DE 220 OHMIOS

RESISTENCIA DE 10 KILO OHMIOS

INGREDIENTES

INTERFACE DE NAVE ESPACIAL

SU PLACA ARDUINO VA A PROTAGONIZAR UNA PELÍCULA DE CIENCIA FICCIÓN

Descubra: entrada y salida digital, su primer programa, las variables

Tiempo: **45 MINUTOS**

Nivel: **bajo**

Proyecto en el que se basa: **1**

Ahora que tiene los fundamentos de la electricidad bajo control, es el momento de pasar a controlar cosas con su Arduino. En este proyecto, va a construir algo que podría ser una interface de una nave espacial de una película de ciencia ficción de los años 70. Va a montar un panel de control con un pulsador y luces que se encienden cuando presiona el pulsador. Puede decidir que indican las luces "Activar hiper-velocidad" o "¡Disparar los rayos laser!". Un diodo LED verde permanecerá encendido hasta que pulse el botón. Cuando Arduino reciba la señal del botón pulsado, la luz verde se apaga y se encienden otras dos luces que comienzan a parpadear.

Los terminales o pins digitales de Arduino solo pueden tener dos estados: cuando hay voltaje en un pin de entrada y cuando no lo hay. Este tipo de entrada es normalmente llamada digital (o algunas veces binaria, por tener dos estados). Estos estados se refieren comúnmente como **HIGH (alto)** y **LOW (bajo)**. **HIGH** es lo mismo que decir "¡aquí hay tensión!" y **LOW** indica "¡no hay tensión en este pin!". Cuando pone un pin de **SALIDA (OUTPUT)** en estado **HIGH** utilizando el comando llamado **digitalWrite()**, está activandolo. Si mide el voltaje entre este pin y masa, obtendrá una tensión de 5 voltios. Cuando pone un pin de **SALIDA (OUTPUT)** en estado **LOW**, está apagandolo.

Los pins digitales de Arduino pueden trabajar como entradas o como salidas. En su código, los configurará dependiendo de cual sea su función dentro del circuito. Cuando los pins se configuran como salidas, entonces podrá encender componentes como los diodos LEDs. Si se configuran como entradas, podrá verificar si un pulsador está siendo presionado o no. Ya que los pins 0 y 1 son usados para comunicación con el ordenador, es mejor comenzar con el pin 2.

MONTANDO EL CIRCUITO

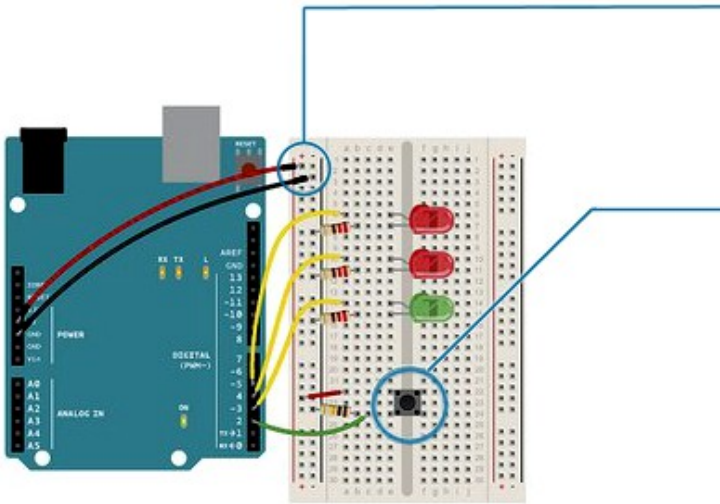


Figura 1

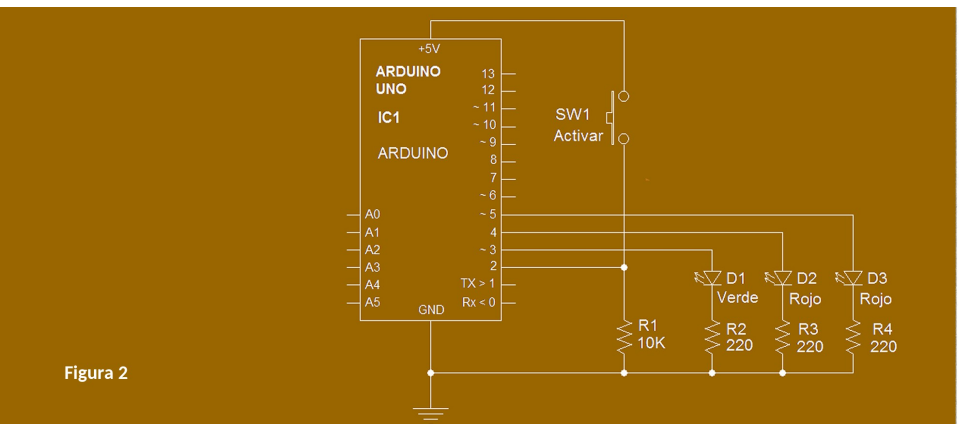
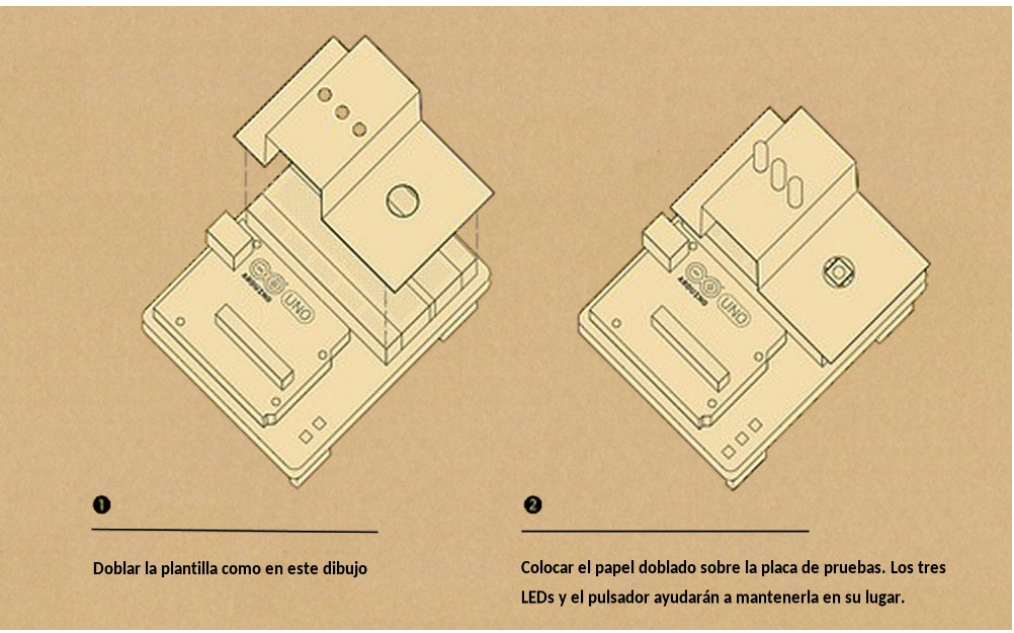


Figura 2

1. Conectar la placa de pruebas a las conexiones de 5V y masa de Arduino, igual que en el proyecto anterior. Colocar los dos diodos LED rojos y el LED verde sobre la placa de pruebas. Conectar el cátodo (patilla corta) de cada LED masa a través de una resistencia de 220 ohmios. Conectar el ánodo (patilla larga) del LED verde al pin 3 de Arduino. Conectar los ánodos de los LEDs rojos a los pins 4 y 5 respectivamente.

2. Colocar el pulsador sobre la placa de pruebas como hizo en el proyecto anterior. Conectar un extremo a la alimentación y el otro terminal del pulsador al 2 de Arduino. También necesita añadir una resistencia de 10K ohmios desde masa al pin del interruptor que va conectado a Arduino. Esta resistencia de puesta a cero conecta el pin a masa cuando el pulsador está abierto, así que Arduino lee LOW cuando no hay tensión en ese pin del pulsador.

Puede cubrir la placa de pruebas con una plantilla suministrada en el kit. O puede decorarla para fabricar su propio sistema de control de la nave. Las luces que se encienden y se apagan no nos dicen nada, pero cuando se colocan en un panel de control y se le coloca una etiqueta, estas sí que sabemos para que valen. ¿Que quiere que indique la luz verde?, ¿qué significa el parpadeo de los LEDs rojos?, ¡usted decide!



❶ Doblar la plantilla como en este dibujo

❷ Colocar el papel doblado sobre la placa de pruebas. Los tres LEDs y el pulsador ayudarán a mantenerla en su lugar.

EL CÓDIGO

Algunas notas antes de comenzar

Cada programa de Arduino tiene dos funciones básicas principales. Las funciones son partes de un programa de ordenador que ejecuta instrucciones específicas. Las funciones tienen un único nombre y son "llamadas" cuando se necesitan. Estas dos funciones principales en un programa de Arduino son llamadas con **setup()** y **loop()**, la cuales necesitan ser declaradas, esto quiere decir que es necesario indicarle a Arduino lo que estas funciones harán. En la siguiente página y en su parte superior dentro del primer cuadro se puede ver como hay que declarar estas funciones **setup()** y **loop ()**. Lo primero que hay que hacer antes de meterse en la parte principal del programa es crear una variable. Las variables son nombres que se utilizan para guardar información dentro de la memoria de Arduino. Los valores de las variables pueden cambiar dependiendo de la instrucciones que contenga el programa. El nombre de las variables deben de ser una descripción de tipo de información que contienen. Por ejemplo, una variable llamada **SwitchState** (estado del pulsador) le dice lo que está guardando: el estado de un pulsador. Por otra parte, una variable llamada "**x**" no dice mucho acerca del tipo de información que guarda.

Vamos a comenzar a programar

Para crear una variable, es necesario declarar de que *tipo* se trata. Una variable de *tipo* **int** guardará un numero entero (también llamado integer); eso significa que almacena cualquier número sin decimales. Cuando se declara una variable, normalmente también se le asigna a la vez un valor inicial. Las declaraciones de la variables siempre deben de finalizar con un punto y coma (**;**).

Configurando como funciona un pin

La función **setup()** solo se ejecuta una vez, cuando Arduino recibe la alimentación para funcionar. Esta función define un bloque, el cual se abre con una llave "**{**" y se cierra con otra llave "**}**", en donde se escriben las instrucciones que configuran los pins digitales de Arduino como entradas o como salidas, usando para ello una función llamada **pindMode()**. Los pins conectados a los LEDs serán **OUTPUT**s y el pin de un pulsador será una **INPUT**.

Crear la función loop

La función **loop()** se ejecuta continuamente después de que la función **setup()** se haya completado. A través de las instrucciones que se incluyen dentro del bloque después de la función **loop()** es donde se comprobará el voltaje de las entradas y si las salidas están activadas o desactivadas. Para verificar el nivel de voltaje de una entrada digital, se utiliza la función **digitalRead()** la cual comprueba el voltaje en el pin elegido. Para saber que pin debe de verificar **digitalRead()** espera un *argumento*.
Los argumentos son información que se le pasa a las funciones diciéndoles como deben de realizar su trabajo. Por ejemplo, **digitalRead()** necesita un argumento: que pin debe verificar. En el programa de Interface de la Nave Espacial, **digitalRead()** va a verificar el estado del pin 2 para después almacenar el valor dentro de la variable switchState.
Si hay voltaje en el pin 2 cuando **digitalRead()** es llamada, entonces la variable **switchState** almacena un valor **HIGH** (o 1). Si no hay voltaje en el pin 2, **switchState** almacenará el valor **LOW** (o 0).

```
void setup (){
}

void loop (){
}
```

{ Las llaves }

Cualquier código que escriba entre llaves será ejecutado cuando se llama a la función

```
1 int switchState = 0;
```

```
2 void setup (){
3   pinMode(3, OUTPUT);
4   pinMode(4, OUTPUT);
5   pinMode(5, OUTPUT);
6   pinMode(2, INPUT);
7 }
```

Mayúsculas y minúsculas

Poner atención a la hora de escribir en mayúsculas y minúsculas dentro del código. Por ejemplo, **pinMode** es el nombre de una instrucción, pero pinmode producirá un error.

```
8 void loop (){
9   switchState = digitalRead(2);
10  // Esto es un comentario
```

Comentarios

Si alguna vez quiere usar el lenguaje natural dentro del programa, puede escribir un comentario.

Los comentarios son notas que se dejan para recordar lo hace el programa, además el microcontrolador ignora estos comentarios. Para añadir un comentario escribir antes dos líneas inclinadas // y a continuación escribir lo que se desee anotar. El microcontrolador ignorará cualquier texto escrito después de estas dos líneas.

La instrucción if

En la siguiente página en la parte superior derecha se utiliza la palabra “if” para verificar el estado de algo (en este caso el estado del pulsador, en qué posición encuentra). Un estamento **if()** en programación compara dos cosas, y determina si la comparación es verdadera o falsa. Dependiendo de este resultado se lleva a cabo una acción u otra. Cuando en programación se comparan dos cosas hay que usar dos signos de igual ==. Si solo se usa un signo de igual, simplemente se le asigna un valor a una variable en lugar de compararla con algo.

Construyendo su nave espacial

digitalWrite() es la instrucción que permite poner +5V o 0V en un pin de salida. **digitalWrite()** tiene dos argumentos: el número del pin a controlar y que valor se coloca en ese pin, HIGH o LOW. Si quiere apagar los LEDs rojos y encender el LED verde dentro del estamento **if()** el código debería ser como el que se muestra en la siguiente página en la parte superior donde aparecen las tres instrucciones **digitalWrite()**. Con las dos hojas juntas puede verse esta parte del código justo a la derecha de este párrafo.

Si se ejecuta el programa ahora, las luces cambiarán cuando se presione el pulsador. Eso esta bien, pero es posible añadir un poco mas de complejidad al programa para mejorar el efecto final.

En las instrucciones anteriores se le indica a Arduino lo que tiene que hacer cuando el pulsador está abierto. También hay que indicarle que hacer cuando el pulsador está cerrado. El estamento **if()** puede usar opcionalmente el componente **else** que permite hacer otra cosa si la condición inicial no se cumple. En este caso, como se acaba de comprobar a través del código si el pulsador esta **LOW**, hay que escribir nuevas instrucciones para la condición **HIGH** dentro del bloque de la instrucción **else**.

Para conseguir que los LEDs rojos parpadeen cuando el pulsador este presionado es necesario encenderlas y apagarlas dentro del bloque de la instrucción **else** que se acaba de escribir. Para hacer esto, cambiar el código tal y como se puede ver en el recuadro de la derecha (parte inferior de la hoja siguiente).

Ahora el programa hará que los LEDs rojos parpadeen cuando el pulsador está presionado.

Después de colocar los LEDs a un cierto estado, es necesario que Arduino realice una pausa antes de cambiarlos de nuevo a su estado anterior. Si esta pausa no se realiza los diodos parpadearan demasiado rápido y parecerán que alumbran menos, no se vera el parpadeo. Esto sucede porque Arduino ejecuta esta parte del programa **(loop)** miles de veces por segundo y los LEDs se encienden y se apagan tan rápido que no podemos percibirlo, dando la sensación de que no se apagan, solo que disminuyen su luminosidad. La instrucción **delay()** permite que Arduino deje de ejecutar cualquier cosa que este haciendo durante un periodo de tiempo. Dentro del argumento de la instrucción **delay()** se establece el número de milisegundos que Arduino estará parado antes de ejecutar la siguiente parte del código. Hay 1000 milisegundos en un segundo. **delay(250)** producirá una pausa de un cuarto de segundo.

```
11   if (switchState == LOW) {
12   // el pulsador no está presionado
13     digitalWrite(3, HIGH); // LED verde
14     digitalWrite(4, LOW); // LED rojo
15     digitalWrite(5, LOW); // LED rojo
16   }
17   else { // el pulsador está presionado
18       digitalWrite(3, LOW);
19       digitalWrite(4, LOW);
20       digitalWrite(5, HIGH);
21       delay(250); // en pausa un cuarto de segundo
22       // cambiar el estado de los LEDs
23       digitalWrite(4, HIGH);
24       digitalWrite(5, LOW);
25       delay(250); // en pausa un cuarto de segundo
26   }
27 } // Volver al comienzo de la instrucción loop
```

Puede ser útil escribir el flujo del programa en pseudo código: es una forma que describe lo que tiene que hacer el programa en lenguaje plano, pero de una forma que hace mas fácil escribir el código final a partir de este lenguaje plano. En este caso se va a comprobar si **switchState** está **HIGH** (si el pulsador está presionado) o no lo está. Si el pulsador está presionado se apagará el LED verde y los rojos se encenderán. En el recuadro de la izquierda pueden verse las instrucciones de este pseudo código.

Si el switchState esta LOW:
  encender el LED verde
  apagar los LEDs rojos

Si el switchState esta HIGH:
  apagar el LED verde
  encender los LEDs rojos

COMO SE UTILIZA

Una vez que Arduino esta programado, el LED verde debe de verse encendido. Cuando presione el pulsador, los LEDs rojos comenzarán a parpadear y el LED verde se apagará. Intenta cambiar el tiempo de las dos instrucciones **delay()**; observa que le pasa a los LEDs y como la respuesta del sistema ha cambiado al variar la velocidad del parpadeo. Cuando en el programa se llama a la instrucción **delay()** hace que todo deje de funcionar durante un tiempo. No se puede leer la posición del pulsador hasta que el tiempo establecido dentro de la instrucción **delay()** haya finalizado. Los retrasos dentro de un programa son a menudo útiles, pero hay que fijarse al realizar un proyecto que su uso no es innecesario y que no interfiere el funcionamiento de dicho proyecto.

¿Cómo podría conseguir que los diodos LEDs comiencen a parpadear cuando el programa comienza? ¿Cómo puedo hacer una interface para mis aventuras interesterales más grande, o más compleja con LEDs y pulsadores

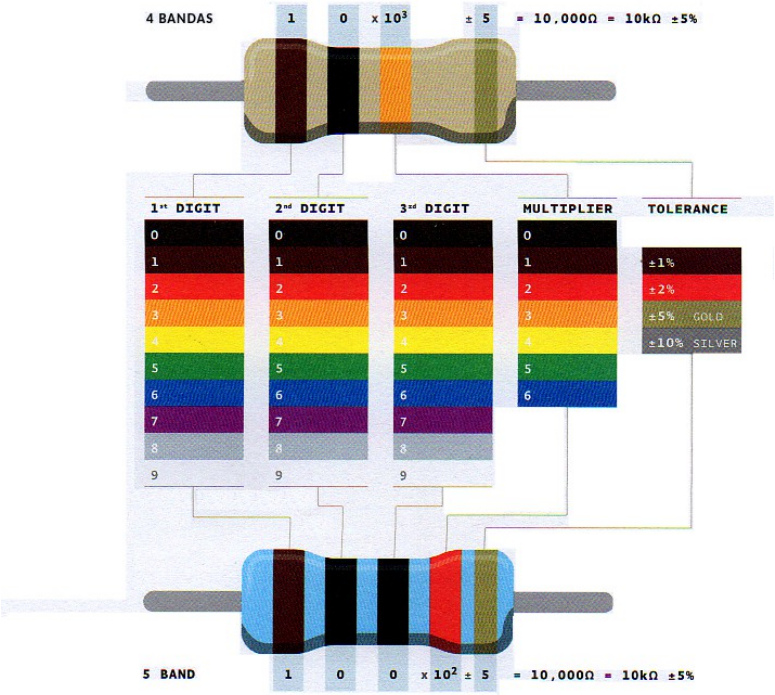
Cuando se inicia la creación de una interface para un proyecto, debemos pensar acerca de las expectativas de las personas mientras lo están usando. Cuando presionen el pulsador, ¿querrán que ocurra algo de inmediato? ¿Debe de existir un tiempo de retraso entre la pulsación y lo que Arduino hace? Intenta ponerte en el lugar de las personas que lo van a usar en el momento de realizar el diseño, y ver si cumple con las expectativas de lo que se espera del proyecto.

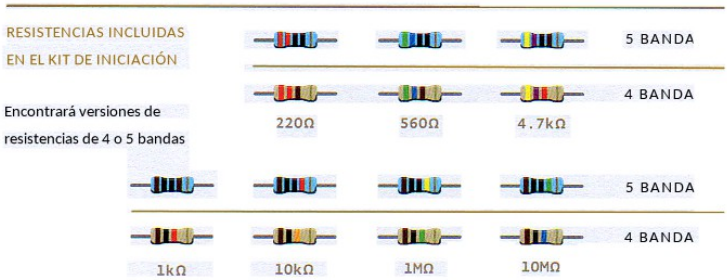
En este proyecto, ha creado su primer programa para Arduino para controlar el comportamiento de algunos LEDs a través de un pulsador. Ha usado variables, una instrucción if()...else, y funciones para leer el estado de una entrada y controlar salidas.

CÓMO LEER EL CÓDIGO DE COLORES DE LAS RESISTENCIAS

Los valores de las resistencias se indican mediante bandas de colores, según un código que se desarrollo en 1920, cuando erá muy difícil escribir números en objetos tan pequeños.

Cada color se corresponde con un número, como se puede ver en la tabla inferior. Cada resistencia tiene entre 4 o 5 bandas. En las resistencias con 4 bandas, las dos primeras bandas indican los dos primeros dígitos del valor de la resistencia, mientras que la tercera banda de color indica el número de ceros que sigue a los dos primeros valores (técnicamente esta tercera banda representa potencias de diez). La última banda especifica la tolerancia: en el ejemplo inferior, se puede leer un valor de resistencia de 10K, y este valor puede variar en más o menos un 5% según esta tolerancia





03

DIODO LED

RESISTENCIA DE 220 OHMIOS

SENSOR DE TEMPERATURA

INGREDIENTES

MEDIDOR DE ENAMORAMIENTO

CONVIERTA SU PLACA ARDUINO EN UNA MÁQUINA QUE MIDE LO ENAMORADO QUE ESTÁ. USANDO UNA ENTRADA ANALÓGICA, VA A PODER REGISTRAR SU NIVEL DE "ENAMORAMIENTO"

Descubra: entrada analógica, usando el monitor serie

Tiempo: **45 MINUTOS**

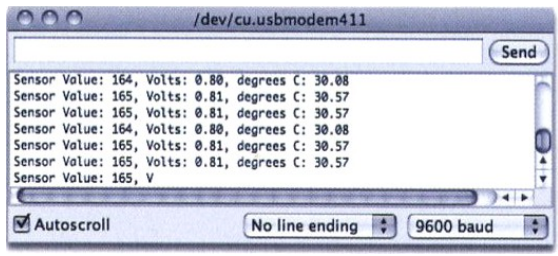
Nivel: **bajo**

Proyectos en los que se basa: **1,2**

Aunque los pulsadores e interruptores son útiles, existen muchas más cosas en el mundo físico que solo encender y apagar algo. Aunque Arduino es una herramienta digital, es posible adquirir con él información de sensores analógicos para medir parámetros físicos como la temperatura o el nivel de iluminación. Para poder hacerlo, Arduino dispone en su interior de un Convertidor Analógico – Digital (ADC), el cual transforma una señal analógica presente en su entrada en una señal digital. Las entradas analógicas de Arduino son los pins A0 – A5 las cuales pueden proporcionar un valor entre 0 y 1023, que equivale a un rango de 0 voltios a 5 voltios, por ejemplo, si la tensión de una de las entradas analógicas vale 2.5V el valor que proporciona el convertidor ADC vale 512.

Va a usar un *medidor de temperatura* para medir la temperatura de la piel. Este componente varía su tensión de salida dependiendo de la temperatura que detecta. Dispone de tres terminales: uno se conecta a masa, otro se conecta a la alimentación y el tercero produce una tensión de salida variable que se aplica a Arduino. En el sketch de este proyecto, se va a leer la tensión de salida del sensor y usarla para encender o apagar unos diodos LEDs, como indicadores de la temperatura de su piel (lo enamorado que está). Existen varios modelos de sensores de temperatura. Este modelo, el TMP36, es utilizado por que los cambios de la tensión de salida son directamente proporcionales a la temperatura en grados Celsius.

El IDE de Arduino incorpora una herramienta llamada *monitor serie* el cual le proporciona información de lo que el microcontrolador está haciendo. Utilizando el monitor serie, se puede conseguir información acerca del estado de sensores, y así tener una idea de lo que sucede en un circuito y en el código cuando se esta ejecutando.



Monitor serie

Figura 1

MONTANDO EL CIRCUITO

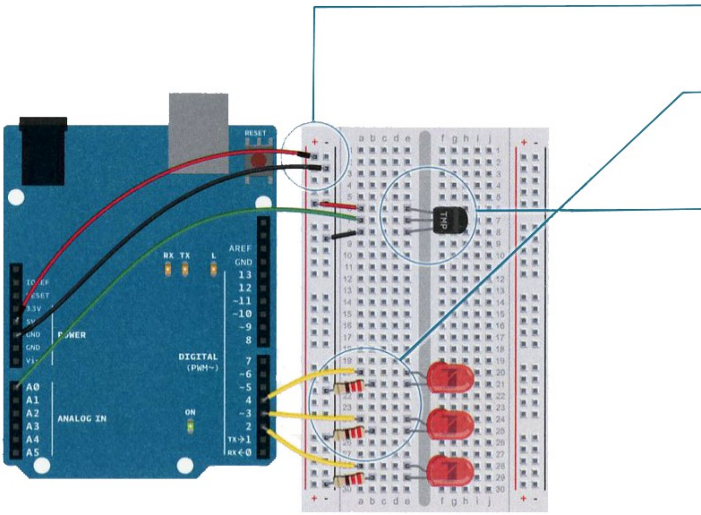


Figura 2

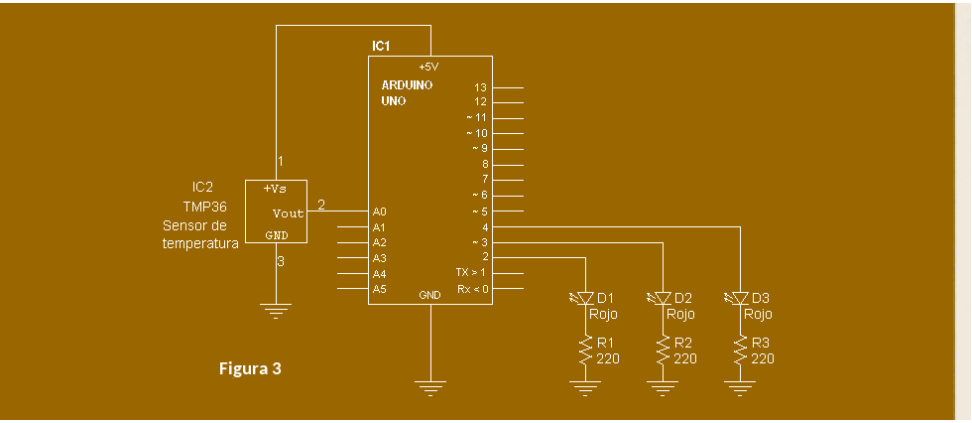


Figura 3

En este proyecto, necesita chequear la temperatura ambiente de la habitación antes de ejecutar el programa. Esto se puede comprobar ahora mismo manualmente, pero también se puede lleva a cabo a través de la calibración que proporciona Arduino, así no será tan complicado. Se puede usar un botón para establecer la temperatura mínima, o establecer dentro del programa este valor antes de que se ejecute el bucle loop() y usarlo como punto de referencia. El proyecto 6 entra en detalle sobre esto, o puede mirar el ejemplo de calibración que se muestra en la siguiente página de Arduino:

arduino.cc/calibration

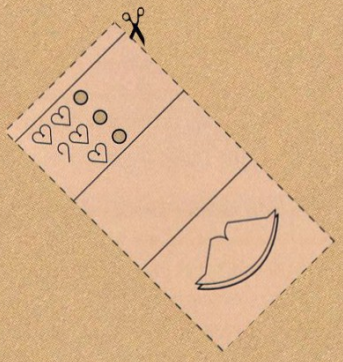
También es posible acceder a otro ejemplo parecido desde el IDE de Arduino dentro de Archivos > Ejemplos > 03Analog > Calibration

1. Tal como lo ha estado haciendo en proyectos anteriores, conectar la placa de pruebas a la alimentación y a masa.

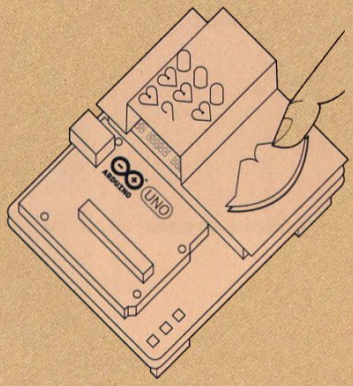
2. Conectar el cátodo (terminal corto) de cada diodo LED a masa a través de una resistencia en serie de 220 ohmios. Conectar los ánodos de los LEDs a los terminales 2, 3 y 4 respectivamente. Estos diodos serán los indicadores de este proyecto.

3. Colocar el sensor de temperatura TMP36 sobre la placa de pruebas con la parte curvada de su cuerpo mirando hacía el lado contrario de la placa Arduino (la colocación de los terminales es importante) como se muestra en la figura 2. Conectar el terminal de la izquierda de la cara plana del cuerpo a la alimentación y el terminal derecho a masa. Conectar el pin central al pin A0 de la placa Arduino. Este pin es la entrada analógica 0.

Crear una interface para el sensor de manera que la gente pueda interactuar con él. Un cartón cortado adecuadamente puede servir perfectamente para esta aplicación. Para realizar un diseño original puede dibujar unos labios en la zona del sensor. También puede etiquetar los LEDs para darles algún significado. Por ejemplo, si solo se enciende un led puede indicar que no está enamorado, dos LEDs encendidos que le gusta alguien, y tres LEDS encendidos querrá decir que está muy enamorado.



1. Corte un pedazo de papel que se ajuste sobre el tablero Dibujar un conjunto de labios donde se coloca el sensor, y cortar los círculos para que los LEDs puedan pasar.



2. Coloque la plantilla de cartón sobre la placa de pruebas de manera que el sensor y los LEDs encajen en los agujeros. Colocar un dedo sobre los labios para comprobar lo enamorado que está.

EL CÓDIGO

Un par de constantes útiles

Las constantes son similares a las variables (guardar información) las cuales permiten usar un único nombre con los datos del programa, pero a diferencia de las variables su contenido no se puede cambiar. Definir una constante con un nombre para la entrada analógica que haga referencia al tipo de pin que se trata (**Pin_del_Sensor**), y definir otra constante con un nombre para la temperatura de referencia (**Temperatura_de_Referencia**). Para cada dos grados sobre esta temperatura de referencia un led se enciende, por ejemplo, si la temperatura de referencia es de 20 grados, con 22 grados se enciende el primer led, con 24 grados el segundo y así sucesivamente. En la lección anterior vimos el tipo de datos INT (página 36 segundo párrafo), usado en este programa para identificar cual es el pin analógico de entrada de Arduino en donde se conecta la salida del sensor de temperatura. Por otro lado el valor de la temperatura de referencia se guarda como una constante en coma flotante. Es tipo de número tiene un punto decimal, y es usado para números que se pueden ser expresados como fracciones o para expresar números con decimales.

Inicializar el puerto serie para establecer la velocidad de comunicación con el ordenador

Dentro del bloque de la configuración “setup()” vamos a usar un nuevo comando, **Serial.begin()**. Este comando permite una comunicación entre la tarjeta Arduino y el ordenador, de manera que pueda ver los valores que se leen en la entrada analógica en la pantalla de un ordenador.
El argumento 9600 de la instrucción Serial.begin establece la velocidad a la que Arduino se comunica con el ordenador, 9600 bits por segundo. Usará el monitor serie del IDE de Arduino para ver los datos que almacena una variable que ha escogido y que es enviada desde el microcontrolador de Arduino. Cuando abra el monitor serie del IDE verificar que la tasa de baudios (bits por segundo) es de 9600.

Definir cuales son las salidas de los pins digitales y ponerlos a cero

La siguiente instrucción **for()** define algunos de los pins como salidas digitales dentro de un bucle. Algunos de estos pins ya se han usado con los LEDs en el proyecto anterior. Ahora en lugar de escribir cada instrucción **pinMode()** para cada uno de estos pins, se utiliza la instrucción **for()**, la cual ejecuta un bucle definiendo las características de cada pin rápidamente. Se trata de una forma de ahorrar tiempo y líneas de código si es necesario definir las propiedades de muchos pins a la vez. La instrucción **for()** se ejecuta en un bucle donde se definen las características de los pins 2 a 4 consecutivamente, definiendo cada pin como una salida digital (**pinMode(NumeroPin, OUTPUT);**) y además en estado bajo (**digitalWrite(NumeroPin; LOW)**).

El siguiente bloque de código es el programa en sí, el cual comienza con la función **loop()**. Aquí se usa una variable llamada **Valor_del_Sensor** en donde se guarda la lectura del sensor. Para poder leer la información del sensor, se utiliza la instrucción **analogRead()** la cual toma un argumento: de que pin se va a tomar la información de lectura del sensor; siendo este argumento la constante **Pin_del_Sensor** que se define al comienzo del programa. El valor leído, el cual esta comprendido entre 0 y 1023, representa indirectamente la tensión que existe en ese pin (para “0” son 0 voltios y para “1023” son +5 voltios).

Enviar los valores del sensor de temperatura al ordenador

La función **Serial.print()** enviá información desde Arduino al ordenador al que está conectado. Puede ver esta información en el monitor serie. Si dentro del argumento de **Serial.print()** escribe una palabra entre comillas, mostrará este texto en el monitor serie tal y como esta escrito, en este caso “**Valor del sensor:** ”. Si le da una variable como argumento, el monitor serie mostrará el valor de esa variable, como se ve a continuación: **Serial.print(Valor_del_Sensor)**, el monitor serie podrá mostrar cualquier valor comprendido entre 0 y 1023.

```
const int Pin_del_Sensor = A0;
const float Temperatura_de_Referencia = 20.0;
```

```
void setup() {
    Serial.begin(9600); // Abrir el puerto serie
```

```
    for(int NumeroPin=2; NumeroPin<5; NumeroPin++){
      pinMode(NumeroPin,OUTPUT);
      digitalWrite(NumeroPin,LOW);
  }
}
```

Para ver un tutorial sobre la instrucción for() abrir este enlace:

arduino.cc/for

```
void loop(){
    int Valor_del_Sensor = analogRead(Pin_del_Sensor);
```

```
    Serial.print("Valor del sensor: ");
    Serial.print(Valor_del_Sensor);
```

Convertir la lectura del sensor a tensión

Usando las matemáticas, es posible averiguar la tensión real que existe en el pin. La tensión será una valor entre 0 y 5 voltios, y podrá tener una parte fraccionaria (por ejemplo, podría ser 2.5 voltios), por eso será necesario almacenar este valor de tensión dentro de una variable con coma **flotante**. Se crea una variable llamada **Tension** en donde se guarda este número. Se divide el valor que almacena la variable **Valor_del_Sensor** por 1024 y se multiplica por 5. El resultado obtenido representa la tensión que existen en el pin.

Al igual que se hizo con el valor del sensor, se mostrará el valor de esta tensión dentro de la ventana del monitor serie.

Convertir la tensión a temperatura y enviar este valor al ordenador

Si examina la hoja de datos del sensor, se muestra información sobre el rango de la tensión de salida. Las hojas de datos son como los manuales de los componentes electrónicos. Son escritas por ingenieros, para otros ingenieros. En la hoja de datos de este sensor se indica que por cada 10 mili voltios de cambio en la tensión del sensor equivale a un cambio de 1 grado Celsiu de temperatura. También indica que el sensor puede leer temperaturas por debajo de los 0 grados. Debido a esto, es necesario crear un offset para valores por debajo de los cero grados. Si se obtiene la tensión, se le resta 0.5 y se multiplica por 100, y así se obtiene la temperatura exacta en grados Celsius. Se almacena este valor dentro de una variable con coma flotante llamada **Temperatura**.

Una vez leída la temperatura real del sensor también se puede mostrar este valor en la ventana del monitor serie. Puesto que la variable de temperatura es lo último que se va a "imprimir" en el monitor serie dentro de este bucle, se va a usar un comando ligeramente diferente: **Serial.println()**. Este comando crea una nueva línea en el monitor serie después de enviar el valor que se va a mostrar. De esta manera se pueden ver mejor los resultados de las medidas al mostrarse los datos en líneas independientes.

Apagar los LEDs para una temperatura baja

Con la temperatura real, ahora se puede configurar la instrucción **if()...else** para encender los LEDs. Utilizando como punto de partida la temperatura de referencia, se encenderá un LED por cada dos grados que se incremente la temperatura sobre la temperatura de referencia. Se va a buscar un rango de valores mientras se mueve a través de la escala de temperatura, esto quiere decir que el segundo LED deberá de encenderse si la temperatura está cuatro grados por encima de la temperatura de referencia, y el tercer LED se encenderá si está sobre seis grados.

```
// Convertir la lectura ADC a tensión
float Tension = (Valor_del_Sensor/1024.0) * 5.0 ;
```

```
Serial.print(", Voltios: ");
Serial.print(Tension);
```

```
Serial.print(", grados C: ");
// Convertir la tensión en temperatura en valores en grados
float Temperatura = (Tension - 0.5) * 100;
Serial.println(Temperatura);
```

Hojas de datos de los componentes electrónicos del Kit de Iniciación

arduino.cc/kitdatasheets

```
if(Temperatura < Temperatura_de_Referencia){
    digitalWrite(2, LOW);
    digitalWrite(3, LOW);
    digitalWrite(4, LOW);
```

Encender un LED para una temperatura baja

El operador && significa “Y” en una sentencia lógica, equivale a la multiplicación. Se utiliza para verificar múltiples condiciones: “Si la temperatura es 2 grados mayor que la temperatura de referencia y si es menor que 4 grados por encima de dicha temperatura de referencia.”

Encender dos LEDs para una temperatura media

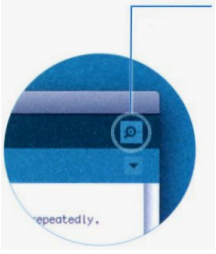
Si la temperatura está entre dos y cuatro grados sobre la temperatura de referencia, este bloque de código también enciende el LED conectado al pin 3.

Encender tres LEDs para una temperatura alta

Apagar los LEDs para una temperatura baja

El convertidor analógico digital puede leer demasiado rápido, así que hay que introducir una pequeña pausa al final del programa (lazo loop()). Si no existiese esta pausa se producirán errores en los valores que se leen.

USARLO



Una vez cargado el programa dentro de la tarjeta Arduino, hacer click sobre el icono del monitor serie. Debe de ver que aparece un listado de valores en el siguiente formato: **Valor del sensor: 200, Voltios; .70, grados C:17**

Coloque sus dedos sobre el sensor mientras está colocado sobre la placa de pruebas y ver que valores aparecen en el monitor serie. Tomar nota de la temperatura que el sensor detecta cuando está al aire.

Cerrar el monitor serie y cambiar el valor de la constante de la temperatura de referencia en el programa al mismo valor que se muestra en el monitor serie cuando el sensor está al aire. Cargar el código de nuevo a la tarjeta Arduino y a continuación poner los dedos sobre el sensor. Como la temperatura aumenta al colocar los dedos, debe de ver como los LEDs se van encendiendo uno a uno. ¡Felicitaciones, el proyecto funciona!,

```
26    }else if(Temperatura >= Temperatura_de_Referencia+2 &&
       Temperatura < Temperatura_de_Referencia+4){
27          digitalWrite(2, HIGH);
28          digitalWrite(3, LOW);
29          digitalWrite(4, LOW);


30    }else if(Temperatura >= Temperatura_de_Referencia+4 &&
       Temperatura < Temperatura_de_Referencia+6){
31          digitalWrite(2, HIGH);
32          digitalWrite(3, HIGH);
33          digitalWrite(4, LOW);


34     }else if(Temperatura >= Temperatura_de_Referencia+6){
35           digitalWrite(2, HIGH);
36           digitalWrite(3, HIGH);
37           digitalWrite(4, HIGH);


38     }
39          delay(100);
40  }
```

Crear un interface para que dos personas puedan comprobar la compatibilidad entre ambos. Debe de decidir de que tipo de compatibilidad se trata, y como actuarán sobre el sensor. ¿Quizás deberán de juntar sus manos para generar el suficiente calor? ¿Tal vez tengan que abrazarse? ¿Que piensa?

Aumentando los tipos de entradas que puede leer y usando la instrucción analogRead() junto con el monitor serie ha podido ver lo que ocurre dentro de su Arduino. Ahora ya puede usar un mayor número de sensores analógicos y entradas.

04

DIODO LED RGB

RESISTENCIA DE 220 OHMIOS

RESISTENCIA DE 10 KILO OHMIOS

FOTO RESISTENCIA - LDR

PAPEL CELOFÁN

INGREDIENTES

LÁMPARA DE MEZCLA DE COLORES

USANDO UN DIODO LED TRI-COLOR Y TRES FOTO RESISTENCIAS, VA A CREAR UNA LÁMPARA QUE CAMBIA SUAVEMENTE LOS COLORES DEPENDIENDO DE LAS CONDICIONES DE LA ILUMINACIÓN AMBIENTE

Descubra: salida analógica, cambio de escala de rango

Tiempo: **45 MINUTOS**

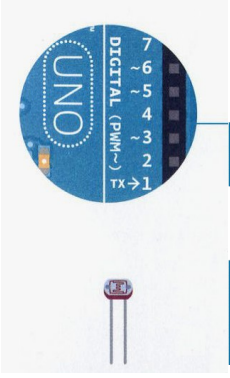
Nivel: **bajo-medio**

Proyectos en los que se basa: **1,2,3**

Hacer que los diodos LEDs parpadeen puede ser divertido, pero ¿se pueden usar para atenuar suavemente la luz que emiten o para producir una mezcla de colores? Se podría pensar que es solo cuestión de disminuir la tensión que se le aplica a un LED para conseguir que su luz se desvanezca.

Arduino no puede variar la tensión de salida de sus pins, solo puede suministrar 5V. Por lo tanto es necesario usar una técnica llamada ***Modulación por Ancho de Pulso o PWM*** (en inglés) para desvanecer suavemente la luz de los LEDs. PWM consigue que un pin de salida varíe rápidamente su tensión de alto a bajo (entre 5 y 0V) durante un periodo fijo de tiempo. Este cambio de tensión se realizada tan rápido que el ojo humano no puede verlo. Es similar a la forma en las que se proyectan las películas de cine, se pasan rápidamente un número de imágenes fijas durante un segundo para crear la ilusión de movimiento.

Cuando la tensión del pin cambia rápidamente de alto (**HIGH**) a bajo (**LOW**) durante un periodo fijo de tiempo, es como si se pudiese variar el nivel de la tensión de ese pin. El porcentaje de tiempo que un pin está en estado **HIGH** con respecto al tiempo total (periodo) se conoce con el nombre de *relación cíclica*. Cuando el pin esta en **HIGH** durante la mitad del periodo y en estado **LOW** durante la otra mitad, la relación cíclica es del 50%. Una relación cíclica baja hace que el diodo LED tenga una luz mucho mas tenue que con una relación cíclica más alta.



La tarjeta Arduino Uno dispone de seis pins que se pueden usar con PWM (***los pins digitales 3,5,6,9,10 y 11***), los cuales se pueden identificar por el símbolo ~ que aparece junto a su número sobre la tarjeta.

Para las entradas en este proyecto, se usarán foto resistencias (sensores que cambian su resistencia dependiendo de la cantidad de luz que llegue a su superficie, también se conocen con el nombre de foto células o resistencias dependientes de la luz LDR). Si conecta uno de sus pin a Arduino, se puede medir el cambio de resistencia al analizar los cambios de tensión en ese pin.

MONTANDO EL CIRCUITO

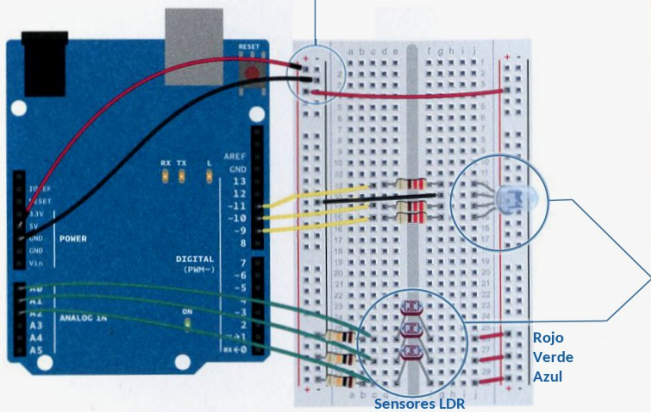


Figura 1

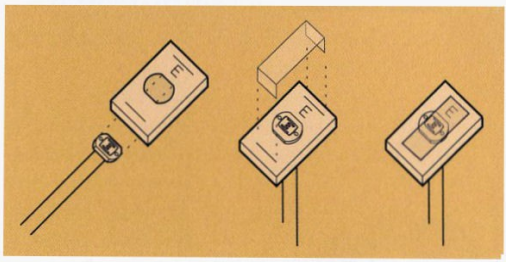


Figura 2

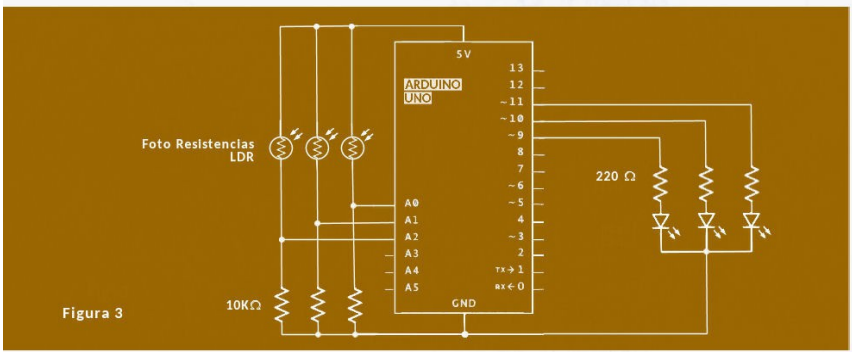


Figura 3

1

Tal como lo ha estado haciendo en proyectos anteriores, conectar la placa de pruebas a la alimentación y a masa.

2

Colocar las tres foto resistencias en el centro que divide la placa de pruebas, como se muestra en la figura 1. Conectar un terminal o pin de cada foto resistencia directamente al positivo de la alimentación. El otro extremo se conecta a una resistencia de 10 Kiloohmios la cual se conecta a masa. Esta resistencia está en serie con la foto resistencia y juntos forman un divisor de tensión. La tensión que aparece en el punto de unión de estos dos componentes es proporcional a la relación entre sus resistencias, según la Ley de Ohm (ver el proyecto número 1 para saber más sobre la Ley de Ohm). Como el valor de la foto resistencia cambia cuando la luz incide en ella, la tensión en este punto de unión también cambia. Conectar el punto de unión entre la foto resistencia y la resistencia de 10K al pin de entrada analógico A0, A1 y A2 respectivamente usando un cable para realizar la conexión.

3

Coger las tres láminas de papel celofán y colocar cada una sobre cada foto resistencia. Colocar el celofán rojo sobre la foto resistencia conectada a la entrada A0, el verde sobre la que se conecta a la entrada A1, y el azul sobre la conectada a la entrada A2. Cada uno de estos papeles de colores actúan como filtros de luz y solo dejan pasar una determinada longitud de onda (color) hacía el sensor sobre el que están encima. El papel de color rojo solo deja pasar la luz roja, el papel verde solo deja pasar la luz verde, y el papel azul solo deja pasar la luz azul. Esto permite detectar los niveles relativos de color de la luz que reciben los sensores.

4

El LED con cuatro pins es un diodo LED RGB de cátodo común. Este diodo LED tiene los elementos rojo, verde y azul separados en su interior, y un masa común (el cátodo). Al aparecer una diferencia de tensión entre el cátodo y las tensiones de salida de los pins de Arduino en formato PWM (los cuales se conectan a los ánodos del LED a través de unas resistencias de 220 ohmios), será posible hacer que el LED varíe la iluminación de sus tres colores. Observar que el pin más largo del LED, insertado en la placa de pruebas, se conecta a masa (ver Figura 1). Conectar los otros tres pins del LED a los pins digitales 9,10 y 11 de Arduino y en serie con cada uno de ellos una resistencia de 220 ohmios. Asegurarse que cada cable del LED se conecta correctamente a su pin PWM correspondiente, tal y como se puede ver en la Figura 1 de la izquierda.

EL CÓDIGO

Un par de constantes útiles

Definir como variables de tipo constante y entero los pins que se van a usar para las entradas y las salidas, y así saber que LED se enciende según el color de la luz que incide en el sensor correspondiente. Para el sensor que detecta la luz roja el LED de color rojo, el LED verde para el sensor de luz de color verde y el LED azul para el sensor de luz azul.

Variables para guardar las lecturas de los sensores así como el nivel de intensidad de cada luz

Añadir variables para almacenar los valores leídos de los sensores así como otras variables en donde almacenar estos valores después de realizar una conversión para poder variar la luminosidad de los LEDs. Se pueda usar el tipo de variable entero para todas estas variables.

Establecer como funcionan los pins digitales y configurar el puerto serie

Dentro del apartado de configuración del programa "**setup()**" establecer la comunicación serie a 9600 baudios por segundo. Como en el ejemplo anterior, se usa esta opción para ver los valores que se leen de los sensores de luz y que se mostrarán en el monitor serie. Además será posible ver un mapa de valores, los cuales se usarán para variar la luminosidad de los diodos LEDs. Aquí también se define los pins de los LEDs como salidas mediante la instrucción **pinMode()**.

Leyendo el valor de cada sensor de luz

En la parte de ejecución del bloque del programa en sí "**loop()**" se leen los valores de los sensores en las entradas analógicas A0, A1 y A2 mediante la instrucción analogRead() y a continuación se almacenan estos valores en las variables que comienzan con el nombre "ValorSensor...". Se coloca una instrucción de retraso "**delay()**" entre cada instrucción de lectura analógica "**analogRead()**" para que el convertidor analógico-digital (ADC) pueda realizar su trabajo.

Informe de las lecturas del sensor de luz al ordenador

Imprimir los valores de los sensores en una sola línea.
El texto "\t" equivale a presionar la tecla "tab" del teclado para realizar una tabulación al comienzo de la línea.

```
const int PinLedVerde = 9;
const int PinLedRojo = 11;
const int PinLedAzul = 10;

const int PinEntradaLDR_Rojo = A0;
const int PinEntradaLDR_Verde = A1;
const int PinEntradaLDR_Azul = A2;
```

```
int ValorSensorRojo = 0;
int ValorSensorVerde = 0;
int ValorSensorAzul = 0;

int ValorRojo = 0;
int ValorVerde = 0;
int ValorAzul = 0;
```

```
void setup() {
 Serial.begin(9600);

 pinMode(PinLedVerde,OUTPUT);
 pinMode(PinLedRojo,OUTPUT);
 pinMode(PinLedAzul,OUTPUT);
}
```

```
void loop() {
 ValorSensorRojo = analogRead(PinEntradaLDR_Rojo);
 delay(5);
 ValorSensorVerde = analogRead(PinEntradaLDR_Verde);
 delay(5);
 ValorSensorAzul = analogRead(PinEntradaLDR_Azul);
```

```
 Serial.print("Mapa de valores sensores \t Rojo: ");
 Serial.print(ValorSensorRojo);
 Serial.print("\t Verde: ");
 Serial.print(ValorSensorVerde);
 Serial.print("\t Azúl: ");
 Serial.print(ValorSensorAzul);
```

Convertir las lecturas de los sensores

La instrucción que cambia el brillo del diodo LED a través de PWM se llama **analogWrite()**. Necesita dos argumentos, el pin sobre el que se escribe y un valor comprendido entre 0 y 255. Este segundo número representa la relación cíclica que aparecerá en el pin que se especifique como salida en Arduino. Un valor de 255 colocará en estado alto (**HIGH**) el pin de salida durante todo el tiempo de esta señal, haciendo que el diodo LED que esta conectado a este pin brille con su máxima intensidad luminosa. Para un valor de 127 colocará el pin a la mitad del tiempo que dura la señal (periodo), haciendo que el diodo LED brille menos que con 255. O se puede fijar el pin en estado bajo (**LOW**) durante todo el periodo, de manera que el diodo LED no alumbrará. Los valores de lectura del sensor de 0 a 1023 hay que convertirlos a valores de 0 a 255 para trabajar dentro del rango de PWM, como se menciona al comienzo de este párrafo. Por tanto los valores de lectura del sensor hay que dividirlos por 4 y los valores convertidos entre 0 y 255 se utilizarán con la instrucción **analogWrite()** para variar el brillo de los diodos LEDs.

Informe de los niveles de iluminación cálculados para los diodos LEDs

Mostrar en el monitor serie un mapa de valores después de dividir por 4 las lecturas de los sensores

Establecer los niveles de iluminación para los LEDs

COMO SE UTILIZA

Una vez que Arduino ha sido programado y conectado, abrir el monitor serie. El diodo LED probablemente mostrará un color blanco apagado, dependiendo del color de la luz predominante en la habitación donde este montado este proyecto. Fijarse en los valores de los sensores que aparecen en el monitor serie, si se encuentra en un entorno con una iluminación que no varía, los números que aparecen apenas van a variar, se mantendrán en valores bastantes constantes.

Apagar la luz de la habitación en donde está el circuito con Arduino montado y ver como varían ahora los valores de los sensores en el monitor serie. Usando una linterna, iluminar cada sensor (LDR) individualmente y observar como los valores cambian en el monitor serie, además de ver como cambia el color que emite el diodo LED. Cuando se cubre las foto resistencias con papel celofán de color, estos sensores solo reaccionarán a una luz de una determinada longitud de onda, es decir, que solo son sensibles a un tipo de color en consonancia con el color del papel celofán que tienen encima. De esta forma se consigue cambiar cada uno de los colores que emite el diodo LED independientemente.

```
31  ValorRojo = ValorSensorRojo/4;
32  ValorVerde = ValorSensorVerde/4;
33  ValorAzul = ValorSensorAzul/4;
```

```
34  Serial.print("Mapa de valores de los sensores \t Rojo: ");
35  Serial.print(ValorRojo);
36  Serial.print("\t Verde: ");
37  Serial.print(ValorVerde);
38  Serial.print("\t Azul: ");
39  Serial.print(ValorAzul);
```

```
40  analogWrite(PinLedRojo, ValorRojo);
41  analogWrite(PinLedVerde, ValorVerde);
42  analogWrite(PinLedAzul, ValorAzul);
43 }
```

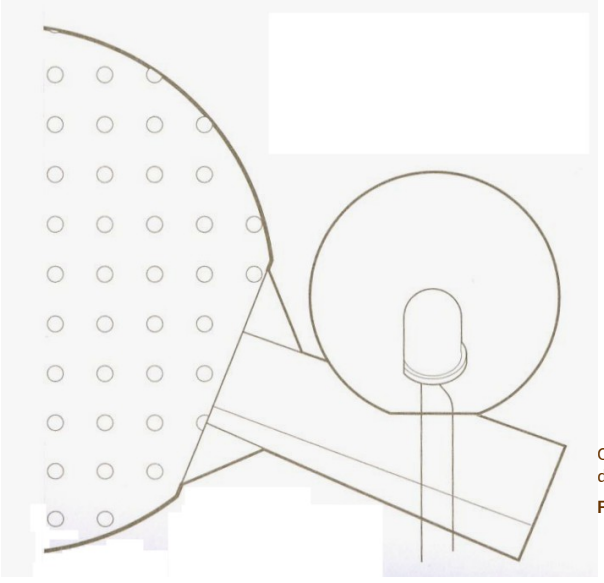
Puede fijarse que la salida de la foto resistencia no trabaja dentro de todo el rango de 0 a 1023. Esto esta bien en este proyecto, pero para obtener una explicación más detallada sobre como calibrar el rango de lectura del sensor ver el proyecto número 6.

Probablemente se habrá fijado que la variación del brillo del diodo LED no es lineal. Cuando el LED esta a la mitad del valor de PWM a 127 (mitad del brillo), aparece mucho menos brillante. Esto es debido a que nuestros ojos no perciben las variaciones del brillo linealmente. El brillo del LED no solo depende del valor de la instrucción **analogWrite()**, sino también de la distancia de la luz desde el difusor, la distancia de los ojos a la luz, y el brillo del LED en relación con otra luz que pueda haber en la habitación.

¿Como podría usar este proyecto para saber si en el exterior hace un buen día mientras usted está trabajando en el interior? ¿Que otros tipos de sensores, en lugar de las resistencias LDR, se pueden usar para controlar los colores del diodo LED?

Aunque el diodo LED no deja de ser un lámpara, la luz que produce nada tiene que ver con la luz de una lámpara. Existen varias formas de difuminar la luz del diodo LED para que se parezca a la luz que produce una bombilla de incandescencia. Realizar un agujero en una pelota de ping pong (figura 4)para colocarla encima del diodo LED y de esta forma difuminar la luz que produce. Otra forma de hacerlo es colocando pegamento transparente sobre el LED o lijando su superficie. No importa que procedimiento utilice, se va a perder un poco de brillo cuando se difumine la luz, pero al final probablemente obtendrá una luz mucho más agradable.

Ya no solo se limita a encender o apagar luces, sino que ahora tiene el control sobre como variar la iluminación de una luz. analogWrite() es la función que permite a los componentes conectados a los pines PWM 3, 5, 6, 9, 10 o 11, variar la relación cíclica de la señal aplicados a ellos.

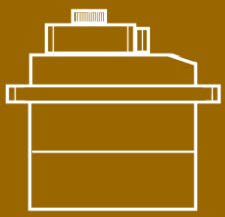


Cortar una bola de ping pong para colocarla encima del diodo led

Figura 4

05

POTENCIÓMETRO



SERVOMOTOR

BRAZO DEL MOTOR

CONDENSADOR DE 100UF

TIRA DE PINS MACHO

INGREDIENTES

Traducido by Tino Fernández Cueto

http://www.futureworkss.com

INDICADOR DEL ESTADO DE ÁNIMO

USAR UN SERVOMOTOR PARA REALIZAR UN MEDIDOR MECÁNICO PARA INDICAR EL ESTADO DE HUMOR QUE TIENE EN ESE DÍA

Descubra: mapa de valores, servomotores, utilización de librerías incorporadas

Tiempo: **1 HORA**

Nivel: **bajo-medio**

Proyectos en los que se basa: **1,2,3,4**

Los servomotores son un tipo especial de motores que no giran alrededor de un círculo continuamente, sino se mueven a una posición específica y permanecen en ella hasta que se les diga que se muevan de nuevo. Los servos solo suelen girar 180 grados (la mitad de un círculo). Combinando uno de estos motores con un pequeño cartón hecho a mano, será capaz de decirle a la gente si deberían venir y preguntar si necesita o no ayuda para su próximo proyecto.

De la misma forma que se ha usado PWM y los LEDs en el proyecto anterior, los servomotores necesitan un número de impulsos para saber que ángulo deben de girar. Los impulsos siempre tienen los mismos intervalos (periodo), pero el ancho de estos impulsos puede variar entre 1000 y 2000 micro segundos. Si bien es posible escribir código para generar estos impulsos, el software de Arduino incluye una librería que permite controlar el motor con facilidad.

Ya que el servomotor solo gira 180 grados y la entrada analógica varía de 0 a 1023, es necesario usar una función llamada **map()** para cambiar la escala de los valores que produce el potenciómetro.

Una de las grandes cosas que tiene la comunidad de Arduino es el talento de la gente quien aumenta su funcionalidad a través de software adicional. De esta forma es posible que cualquiera escriba librerías para aumentar la funcionalidad de Arduino. Existen librerías para una gran variedad de sensores y actuadores y otros dispositivos con la que los usuarios han contribuido en esta comunidad. El software libre extiende la funcionalidad de un entorno de programación. El software de Arduino viene con un número de librerías que son útiles para trabajar con el hardware o lo datos. Una de las librerías incluida esta diseñada para ser usada con servomotores. Cuando se escriba el código será necesario importar esta librería y así se podrá controlar el servomotor.

MONTANDO EL CIRCUITO

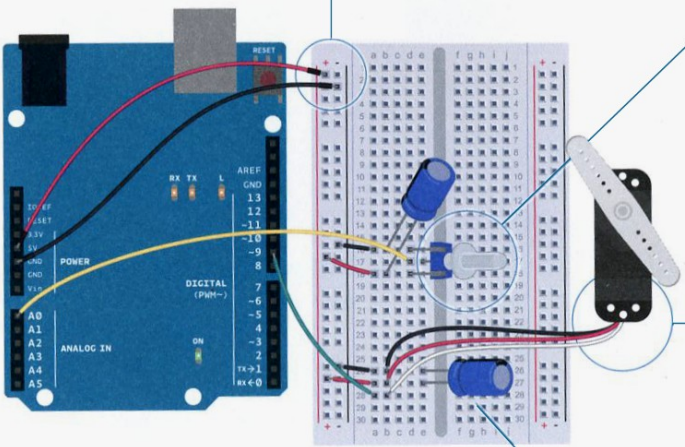


Figura 1

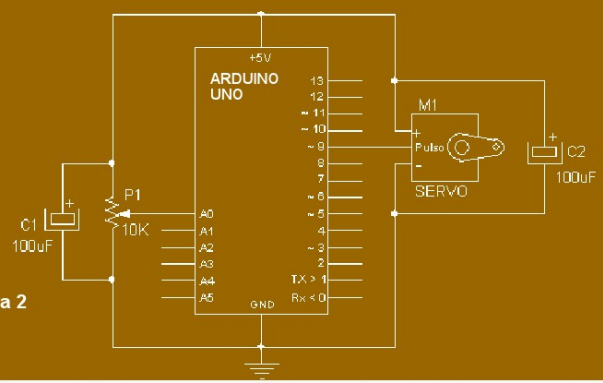
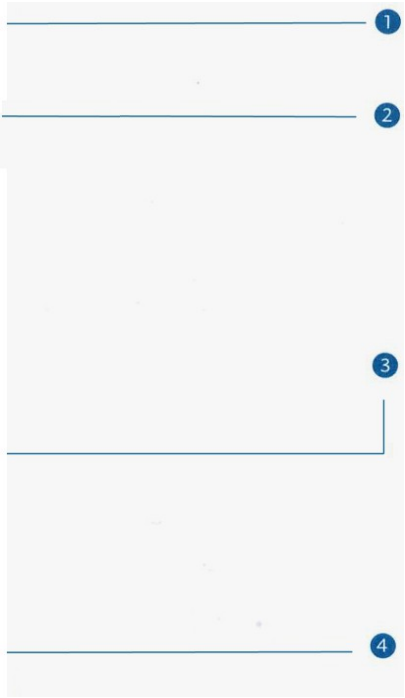


Figura 2

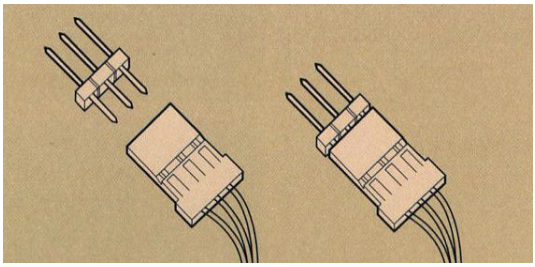


1. Conectar la tensión de +5V y la masa a un lado de las tiras verticales de la placa de pruebas desde la placa de Arduino.

2. Conectar el potenciómetro sobre la placa de pruebas y conectar uno de los pins extremos a +5V y el otro extremo a masa. Un potenciómetro es un tipo de divisor de tensión. Cuando se gira el mando del potenciómetro, se cambia la relación de tensión entre el terminal central y el terminal conectado al positivo de alimentación. Es posible leer el valor de este cambio usando una entrada analógica de Arduino. Conectar por tanto el terminal central a la entrada analógica A0. De esta forma se controlará la posición del servomotor.

3. El servo tiene tres cables que salen de su interior. Uno de ellos es la alimentación (color rojo), otro es la masa (color negro), y el tercero (blanco) es el cable de control a través del cual recibirá la información desde Arduino. Conectar tres pins macho de la tira de pins al conector hembra de los cables del servomotor (ver figura 3). Conectar estos pins machos a la placa de prueba de manera que cada pin se conecte a una fila diferente. Conectar la alimentación de +5V al cable rojo la masa al cable negro y el cable blanco al pin 9 de Arduino.

4. Cuando el servomotor empieza a moverse, consume mucha más corriente que si ya estuviese moviéndose. Esto producirá una pequeña caída de tensión en la placa. Si se coloca un condensador de 100uF entre el positivo y la masa cerca del conector del servomotor como se muestra en la figura 1, se podrá atenuar cualquier caída de tensión que se pueda producir. También se puede colocar un condensador de la misma capacidad entre los extremos del potenciómetro, entre positivo y la masa. Estos condensadores son llamados condensadores de desacoplo porque reducen, o desacoplan, los cambios causados en la línea de alimentación por los componentes del resto del circuito. Hay que tener mucho cuidado al conectar estos condensadores ya que tienen polaridad. Hay que fijarse que uno de los pins del condensador esta indicado con un signo menos (-), por tanto deberá conectarse a masa y el otro pin a positivo. Si se equivoca y coloca algún condensador con la polaridad contraria podría explotar.



El servomotor viene con un conector hembra, así que será necesario añadirle un conector macho de tres pins para poder conectarlo a la placa de pruebas.

Figura 3

EL CÓDIGO

Importar librería

Para usar la librería del servomotor primero tiene que importarla. Esto añade nuevas funciones al sketch desde la librería.

Creación del objeto Servo

Para referirse a la librería servo, es necesario crear un nombre de esta librería en una variable (**MiServo**). Esto se conoce con el nombre de un ***objeto.*** Cuando se hace esto, se crea un nombre único que tendrá todas las funciones y capacidades que la librería servo posee. A partir de este punto en el programa, cada vez que se utilice el nombre **MiServo**, se podrá usar todas esas funciones y capacidades de la librería servo.

Declaración de variales

Poner una variable de tipo constante entero con un nombre para el pin que se une al potenciómetro (**PinPot**), y nombres de variables de tipo entero para guardar el valor de la entrada analógica (**ValorPot**) y el valor del ángulo (**Angulo**) al que el servomotor se debe de mover.

Asociar el objeto Servo con el pin de Arduino e inicializar el puerto serie

Dentro de la función **setup()** es necesario decirle a Arduino que pin está unido al servomotor

Se incluye una comunicación serie con el ordenador para verificar los valores del potenciómetro y ver como se convierten estos valores a ángulos para el servomotor.

Lectura del valor del potenciómetro

Dentro de la función **loop()**, se procede a leer la entrada analógica (A0) donde se conecta el terminal central del potenciómetro y a continuación mostrar este valor en el monitor serie y en la pantalla de un ordenador.

Convertir los valores del potenciómetro a valores para el servo

Para crear valores a partir de la entrada analógica que se puedan usar con el servomotor, se puede hacer de una forma muy fácil usando la instrucción **map()**. Esta instrucción trabaja con escalas de números. En este caso cambia la escala de valores entre 0-1023 a valores entre 0-179. Necesita cinco argumentos: el número que debe de ser escalado (en este caso dentro de la variable ValorPot), el valor mínimo de la entrada (0), el valor máximo de la entrada (1023), el valor mínimo de la salida (0) y el valor máximo de la salida (179). El resultado se guarda dentro de la variable Angulo.

Girando el servo

Finalmente, es el momento de mover el servomotor. El comando **MiServo.write(Angulo)** mueve el servomotor al ángulo que se indica dentro de la variable Angulo del argumento de esta instrucción.
Al final de la función **loop()** se coloca un retraso de 100 mili segundos **delay(100)** para que el servo tenga tiempo a moverse a su nueva posición.

```
#include <Servo.h>

Servo MiServo;

int const PinPot= A0;
int ValorPot;
int Angulo;

void setup() {
  MiServo.attach(9);

  Serial.begin(9600);
}

void loop() {
  ValorPot = analogRead(PinPot);
  Serial.print("Posicion del potenciometro: ");
  Serial.print(ValorPot);

  Angulo = map(ValorPot, 0, 1023, 0, 179);
  Serial.print(" , Angulo: ");
  Serial.println(Angulo);

  MiServo.write(Angulo);
  delay(100);
}
```

COMO SE UTILIZA

Una vez que Arduino ha sido programado y alimentado, abrir el monitor serie desde el IDE de Arduino. Deberá de mostrar una serie de valores similares a estos:

Posicion del potenciometro: 137 , Angulo: 23
Posicion del potenciometro: 242 , Angulo: 42

Cuando se gira el potenciómetro se debe de ver como estos números cambian, además también el servomotor debe de girar a una nueva posición. Observar la relación entre el valor numérico de la variable del potenciómetro (Posición del potenciómetro) y el valor del ángulo en el monitor serie (Angulo) con respecto a la posición real del servomotor. Estos valores deberán de estar en concordancia con el giro del potenciómetro, es decir, cuando el potenciómetro esté girado en sentido contrario a las agujas del reloj, en un extremo, el valor valdrá cero y el servomotor estará en una posición, en cambio al girar el potenciómetro hasta el otro extremo, deberá de indicar un valor de 1023 y un ángulo de 179 de manera que el servomotor se mueva 180 grados en el sentido de las agujas del reloj.

Hay que indicar que el uso del potenciómetro como una entrada analógica permite variar el rango completo entre 0 y 1024. Esto los hace muy útiles para probar proyectos que utilicen entradas analógicas.

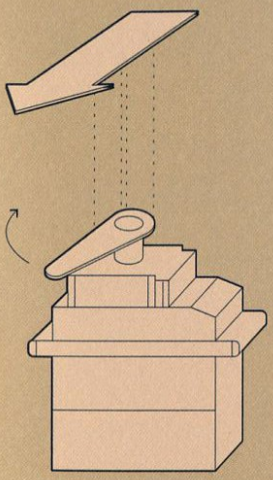
Los servomotores son motores con un número de engranajes y algunos circuitos en su interior. La mecánica del interior proporciona una realimentación al circuito, de manera que sabe siempre cual es su posición. Si bien pueda parecer que tiene limitado su rango de acción, es posible que consiga realizar una gran variedad de diferentes movimientos añadiendo mecanismos adicionales. Hay una serie de recursos que describen este tipo de mecanismos en detalle como ***robives.com/mechs*** y en el libro ***Making Things Move*** de Dustyn Roberts.

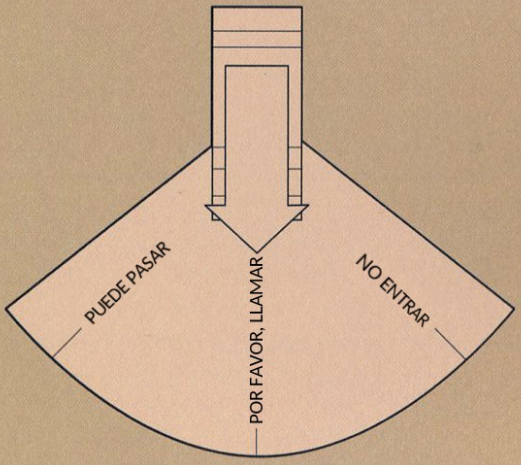
El potenciómetro no es el único sensor que puede usar para controlar la posición del servomotor. Usando la misma configuración física y un sensor diferente (por ejemplo, una flecha que se mueva para indicar varios estados de ánimo), ¿qué clase de indicador se podría realizar? ¿Cómo se podría modificar el proyecto para que trabajará con temperatura (como el medidor de enamoramiento) en lugar de usar un potenciómetro? ¿Podría indicar la hora del día usando una foto resistencia LDR? ¿Cómo habría que cambiar la escala de los valores de estos sensores para que funcionasen con el servomotor?

Los servomotores son fáciles de controlar por Arduino usando una librería, la cual es una colección de código que aumenta las posibilidades del entorno de programación. Algunas veces es necesario para poder usar los valores cambiarlos de una escala a otra.

Ahora que el proyecto esta en marcha y se puede mover, ya puede indicar a las personas si esta dispuesto a ayudarles en sus proyectos o por el contrario prefiere estar solo y que no le molesten.

Con unas tijeras, cortar un trozo de cartón con la forma de una flecha. Colocar el servomotor a 90 grados (verificar el valor del ángulo en el monitor serie sino está seguro). Pegar la flecha de cartón sobre el brazo del servomotor de manera que tenga la misma orientación que el cuerpo. Ahora será posible girar la flecha 180 grados cuando se gire el potenciómetro de un extremo a otro. Disponer de un trozo de papel más grande que el servomotor con la flecha encima y dibujar medio círculo sobre este papel. En uno de los extremos del círculo escribir ***"No entrar"***. En el otro extremo escribir "***Puede pasar"***. En medio del arco escribir "**Por favor llamar**". Colocar el servo con la flecha encima del papel. ¡Felicidades, ahora tiene una manera de decirle a la gente lo que está de ocupado con sus proyectos!





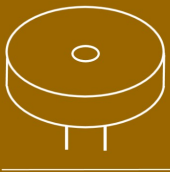
1

Pegar una flecha de papel al brazo del servo.

2

Diseñar una base de papel y colocarla debajo del servo.

06



ZUMBADOR PIEZOELÉCTRICO

FOTO RESISTENCIA - LDR

RESISTENCIA DE 10 KILO OHMIOS

INGREDIENTES

THEREMIN CONTROLADO POR LUZ

¡HA LLEGADO EL MOMENTO DE HACER ALGO DE RUIDO! USANDO UNA FOTO RESISTENCIA Y UN ZUMBADOR, VAMOS A MONTAR UN THEREMIN CONTROLADO POR LA LUZ

Descubra: generar sonido con la función tone(), calibrar sensores analógicos

Tiempo: **45 MINUTOS**

Nivel: **bajo-alto**

Proyectos en los que se basa: **1,2,3,4**

Un Theremin es un instrumento que genera sonidos mediante el movimiento de las manos de un músico alrededor de este instrumento. Probablemente habrá oído alguno de ellos en películas de terror. El Theremin detecta donde se produce el movimiento de la mano del artista en relación a dos antenas al poder leer la variación de la carga capacitiva de estas antenas. Las antenas se conectan a un circuito analógico que crea los sonidos. Una de las antenas controla la frecuencia del sonido y la otra controla el volumen. Aunque Arduino no puede producir de una forma exacta los misteriosos sonidos de este instrumento, si que es posible emularlos usando la función **tone()**. La figura 1 muestra los diferentes tonos que se producen al usar **analogwrite()** y **tone()**. Esto posibilita que un transductor como pueda ser un altavoz o un zumbador se mueva hacia delante y hacia atrás a diferentes velocidades.

Observa como este tono está en estado bajo la mayoría del tiempo, pero la frecuencia es la misma que en el siguiente tono, PWM200.

Observa como en este tono la tensión está en estado alto (+5V) durante la mayoría del tiempo, pero la frecuencia sigue siendo la misma que para el tono anterior, PWM50.

La relación cíclica de este tono es del 50% (la mitad del tiempo esta en estado alto y la otra mitad en estado bajo) y la frecuencia es la misma.

En este tono la relación cíclica es igual que para el tono de 440 pero la frecuencia vale el doble

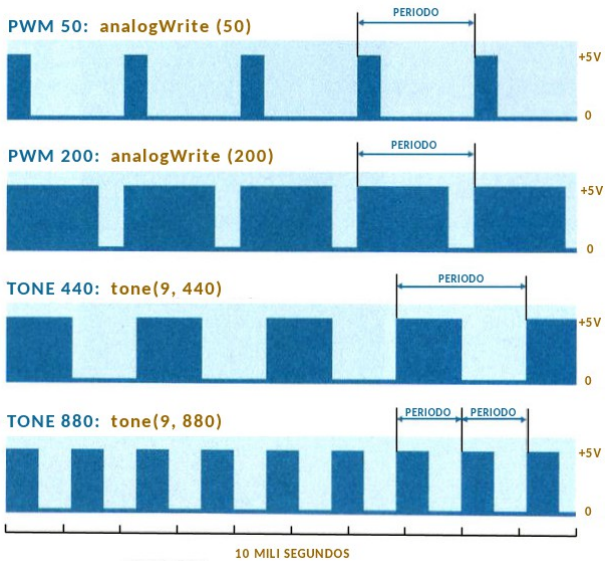


Figura 1

En lugar de usar sensores capacitivos con Arduino, se utilizará una foto resistencia LDR para detectar la cantidad de luz. Al mover las manos sobre el sensor, se variará la cantidad de luz que llega a la superficie de la LDR, como se hizo en el proyecto número 4 (lámpara de mezcla de colores). El cambio de tensión en el pin de una entrada analógica de Arduino (A0) determinará la frecuencia de la nota que se va a escuchar.
La foto resistencia LDR se conectará a Arduino usando un divisor de tensión tal y como se hizo en el proyecto número 4. Probablemente se habrá dado cuenta de que en el proyecto 4 la lectura que se obtenía al usar **analogRead()** no cubría todo el rango de 0 a 1024. La resistencia fija conectada a masa limita la parte baja del rango y el brillo de la luz sobre la LDR limita la parte superior de este rango. En lugar de configurar el circuito para limitar el rango, se calibran las lecturas del sensor entre los valores alto y bajo, convirtiendo estos valores a frecuencias sonoras usando la función **map()** y de esta forma conseguir el mayor rango posible del Theremin.

Un zumbador piezoeléctrico es un componente electrónico que vibra cuando recibe electricidad. Cuando se mueve desplaza el aire a su alrededor, creando ondas de sonido.

MONTANDO EL CIRCUITO

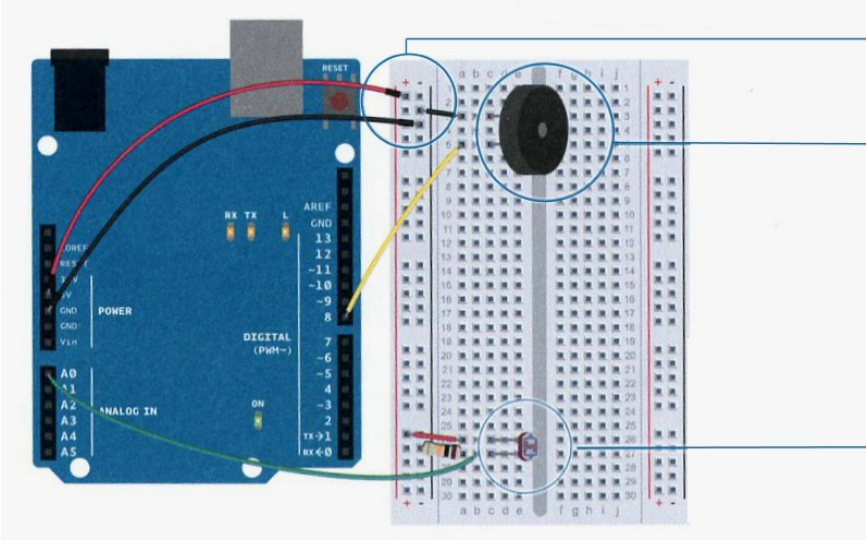


Figura 2

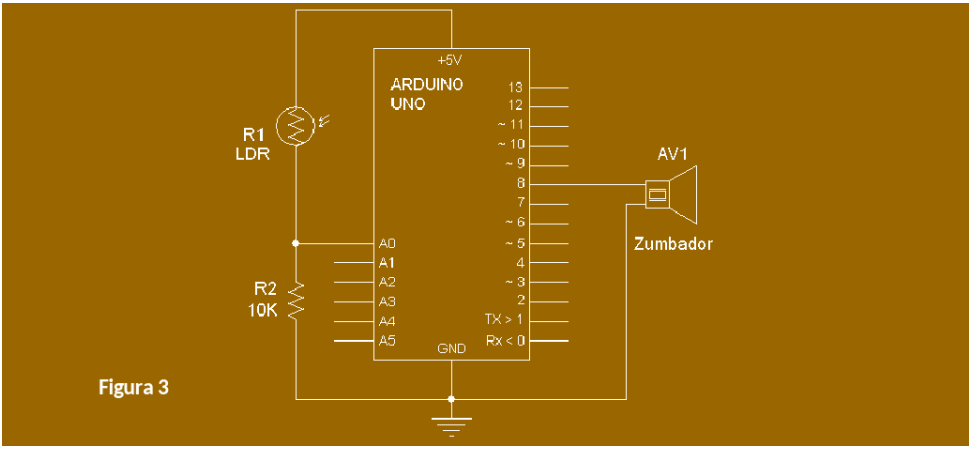


Figura 3

Tradicionalmente el Theremin puede controlar la frecuencia y el volumen del sonido. En este proyecto solamente se podrá controlar la frecuencia. Ya que no se puede controlar el volumen usando Arduino, es posible cambiar manualmente el nivel de tensión que le llega al zumbador. ¿Qué sucede si se conecta un potenciómetro en serie con el zumbador y conectado al pin 8? ¿Y si se colocase otra foto resistencia?

1. Sobre la placa de pruebas, conectar los cables de alimentación y masa desde la placa Arduino.

2. Coger el zumbador y conectar uno de sus pins a masa, el otro pin conectarlo al pin digital 8 de Arduino.

3. Colocar la foto resistencia LDR en la placa de pruebas, conectar uno de sus pins a la columna de alimentación de +5V a través de un cable rojo. Conectar el otro extremo de la LDR al pin analógico A0 de Arduino y a masa a través de una resistencia de 10 Kilo ohmios. Este circuito es el mismo que el divisor de tensión del proyecto número 4.

EL CÓDIGO

Crear variables para calibrar el sensor

Crear una variable (ValordelSensor) que guarde la lectura del valor de la foto resistencia LDR usando la instrucción **analogRead()**. A continuación, crear dos variables, una para almacenar el valor máximo y otra para guardar el valor mínimo (ValorMaximodelSensor y ValoMinimodelSensor). Se establece un valor inicial de 1023 para el valor mínimo y un valor de 0 para el valor máximo. Cuando se ejecute el programa por primera vez, se comparan estos números con los valores obtenidos en la LDR, para encontrar de esta forma los valores máximo y mínimo del rango y poder así calibrar el sensor.

Crear una constante para el indicador de calibración

Crear una constante llamada **PinLed**. Se utilizará para indicar que el sensor está siendo calibrado mientras el diodo LED de la placa Arduino permanezca encendido. Este diodo LED integrado se conecta al pin digital 13 de Arduino.

Establecer el número del pin digital y ponerlo en estado alto

Dentro de la función **setup(),** y usando la instrucción **pinMode(),** definir el pin del led (PinLed) como una salida (**OUTPUT**) y encender este diodo LED.

Usar la instrucción While() para realizar la calibración

Los siguientes pasos calibrarán los valores máximo y mínimo del sensor. Se utiliza la instrucción **while()** para ejecutar otras instrucciones varias veces durante 5 segundos. **while()** ejecuta las instrucciones que contiene durante este tiempo hasta que ciertas condiciones se cumplen. Junto con la instrucción **while()** se utiliza otra instrucción, **mills()**, para saber el tiempo actual. Esta instrucción informa del tiempo que Arduino lleva funcionando desde que se ha sido encendido o desde que se hizo un reset.

Comparar los valores del sensor para realizar la calibración

Dentro de la función **loop()**, se realiza la lectura del valor del sensor; si este valor es mayor que el valor que almacena la variable ValorMaximoSensor (inicialmente 0), se actualiza el valor de esta variable con el valor leído del sensor. Al igual ocurrirá con la variable que almacena el valor minimo (ValorMinimoSensor, inicialmente con un valor de 1024), si el valor leído del sensor es menor esta variable almacenará este valor.

Indicar que la calibración ha finalizado

Cuando los 5 segundos han pasado, la ejecución en bucle de la instrucción while() llegará a su fin. El diodo LED conectado al pin 13 de Arduino dejará de alumbrar, indicando que la calibración ha finalizado. Se utilizarán los valores máximo y mínimo que se acaban de guardar para establecer el rango de frecuencias en la parte principal del programa. Estas frecuencias son las que se van a oír a través del zumbador.

```
1  int ValordelSensor;
2  int ValorMinimoSensor = 1023;
3  int ValorMaximoSensor = 0;
```

```
4  const int PinLed = 13;
```

```
5  void setup() {

6   pinMode(PinLed,13);
7   digitalWrite(PinLed,HIGH);
```

while()
arduino.cc/while

```
8    while(millis() < 5000) {
```

```
9       ValordelSensor = analogRead(A0);
10      if(ValordelSensor > ValorMaximoSensor) {
11       ValorMaximoSensor = ValordelSensor;
12      }
13      if(ValordelSensor < ValorMinimoSensor) {
14       ValorMinimoSensor = ValordelSensor;
15      }
16     }
```

```
17    digitalWrite(PinLed,LOW);
18   }
```

Leer y guardar el valor del sensor

Dentro de la función **loop()**, se procede a la lectura del sensor conectado al terminal A0 de Arduino para almacenarlo dentro de la variable **ValordelSensor**.

Convertir el valor del sensor a frecuencia

Crear una variable llamada **tono**. El valor que se almacena dentro de esta variable **tono** se obtiene de cambiar de rango el valor leído del sensor y que está guardado dentro de la variable **ValordelSensor**. Usar las variables **ValorMaximoSensor** y **ValorMinimoSensor** para establecer los límites del rango que se va a cambiar de escala usando la función **map()**. Los valores iniciales de salida de la nueva escala están comprendidos entre 50 y 4000. Estos números representan el rango de frecuencias que Arduino va a generar a través del zumbador, es decir, producirá frecuencias comprendidas entre 50 y 4000 ciclos por segundo.

Reproducir la frecuencia (tono)

A continuación, se ejecuta la función **tone()** la cual produce un sonido. Esta función necesita de tres argumentos: el número de pin de salida que producirá el sonido (en este caso el pin 8) que frecuencia se va a generar (determinada por la variable tono) y durante cuanto tiempo va a sonar esa nota (probar con 20 mili segundos para comenzar).

Después de ejecutar la función anterior se introduce un retraso de 10 mili segundos para que le de tiempo a Arduino a generar la nota. Se realiza mediante la instrucción **delay(10)**.

COMO SE UTILIZA

Nada mas alimentar la placa Arduino se genera una ventana de 5 segundos para calibrar el sensor; esto se indica por que el diodo LED amarillo de la placa se enciende durante todo este tiempo. En este momento acercar la mano a la foto resistencia, a continuación alejar la mano para no hacer sombra a dicha foto resistencia, de esta manera el programa establece los límites del rango de trabajo en función de la iluminación ambiente. Los movimientos de la mano en el momento de realizar la calibración deberán ser lo más parecidos a los movimientos que se van a realizar después de la calibración para generar los sonidos.

Después de 5 segundos, la calibración estará completada, y el diodo LED de la placa Arduino se apagará. Cuando esto suceda, ¡se debe de escuchar un sonido a través del zumbador piezoeléctrico! Ahora si se mueva la mano sobre la foto resistencia deberá de variar la frecuencia de la nota que se escucha y de esta forma se podrán generar notas con diferentes frecuencias. El efecto final será parecido a la música de fondo que se suele escuchar en las películas de miedo.

```
19 void loop() {
20  ValordelSensor = analogRead(A0);

21  int tono =
     map(ValordelSensor,ValorMinimoSensor,ValorMaximoSensor,50,4000);

22   tone(8,tono,20);

23   delay(10);
24 }
```

El rango que se define dentro de la función **map()** es bastante amplio, se puede intentar modificar estos valores para conseguir que las frecuencias que se generan se ajusten más a su gusto sobre un estilo musical.

La función **tone()** trabaja de una forma muy parecida a PWM cuando se usa con la instrucción **analogWrite(),** pero con la diferencia que al usar esta instrucción la frecuencia de los impulsos generados no cambia: es posible cambiar el ratio de los impulsos durante este periodo de tiempo y así poder variar la relación cíclica sin variar la frecuencia (ver figura 1, PWM 50 y PWM200). Con **tone()** también se envían impulsos pero además se puede cambiar la frecuencia de estos impulsos. La función **tone()** siempre produce impulsos con una relación cíclica del 50%, (la mitad del tiempo la señal está en estado alto y la otra mitad está en estado bajo), puede ver un ejemplo en la figura 1 con los tonos de 440 y 880 (página 71). Con **tone()** no se puede variar la relación cíclica de una señal.

La función tone() ofrece la posibilidad de generar diferentes frecuencias cuando son enviadas a un altavoz o a un zumbador. Cuando se utilizan sensores en un circuito divisor de tensión, no es posible conseguir que los valores varíen todo el rango entre 0 y 1024. Al calibrar los sensores, es posible cambiar estos valores de rango para que se puedan usar.

07

PULSADOR

ZUMBADOR PIEZOELÉCTRICO

RESISTENCIA DE 10 KILO OHMIOS

RESISTENCIA DE 1MEGA OHMIO

RESISTENCIA DE 220 OHMIOS

INGREDIENTES

TECLADO MUSICAL

CON POCAS RESISTENCIAS Y PULSADORES VAMOS A CONTRUIR UN PEQUEÑO TECLADO MÚSICAL

Descubra: circuito mixto con resistencias, matrices

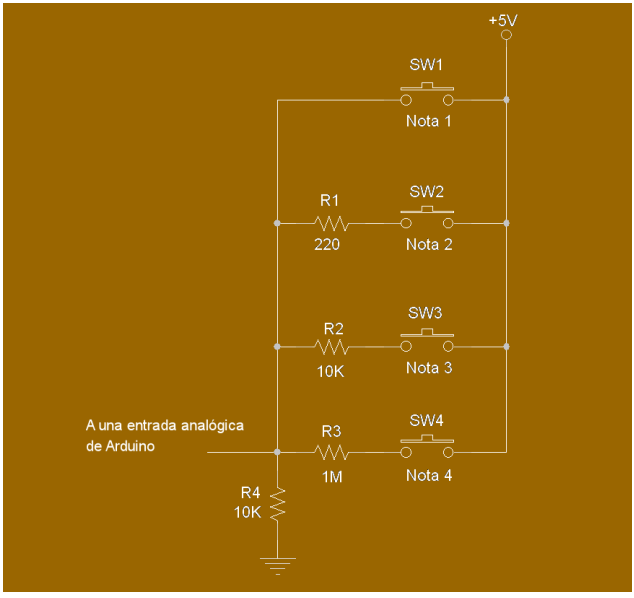
Tiempo: **45 MINUTOS**

Nivel: **medio-alto**

Proyectos en los que se basa: **1,2,3,4,6**

Al utilizar en este proyecto varias resistencias y pulsadores conectados a una entrada analógica de Arduino para generar diferentes tonos, se está construyendo algo que se conoce con el nombre de circuito mixto con resistencias.

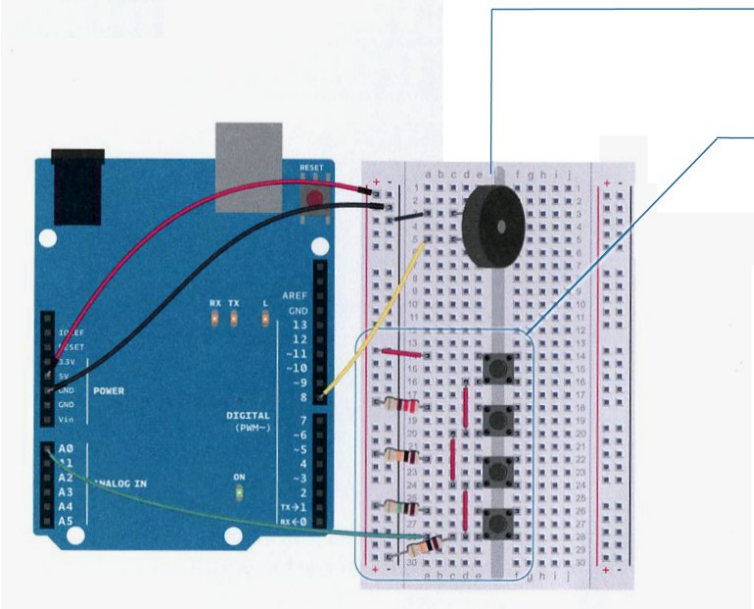
Esta es una forma de leer un número de pulsadores usando una entrada analógica. Es una técnica útil sino entiende como funcionan las entradas digitales. Se conectan un número determinado de pulsadores en paralelo y conectados a la entrada analógica A0 de Arduino. Cada uno de estos pulsadores se conectan al positivo de la alimentación a través de una resistencia. Cuando se presiona un pulsador, aparece una tensión en el terminal de entrada analógico A0 de Arduino, y según que pulsador se presione esta tensión será diferente. Si se presionan dos pulsadores al mismo tiempo se consigue en la entrada analógica un tensión que será proporcional al valor de la resistencias en paralelo de los dos pulsadores presionados. Al final se trata de varios divisores de tensión conectados en paralelo los cuales se activan cada vez que se presione un pulsador.



Circuito mixto con resistencias como entrada analógica.

Figura 1

MONTANDO EL CIRCUITO



La disposición de resistencias y los pulsadores conectados a esta entrada analógica se conoce con el nombre de circuito mixto

Figura 2

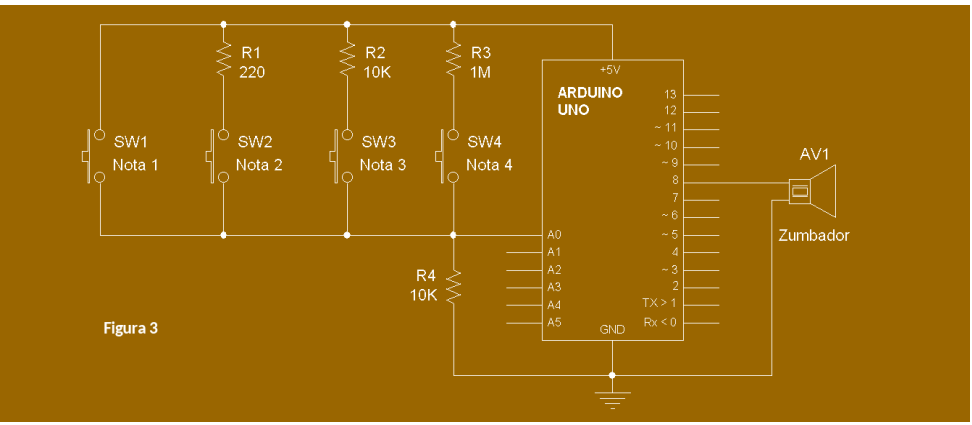
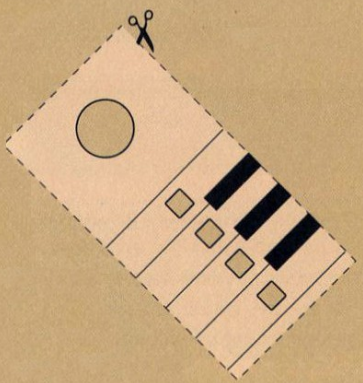


Figura 3

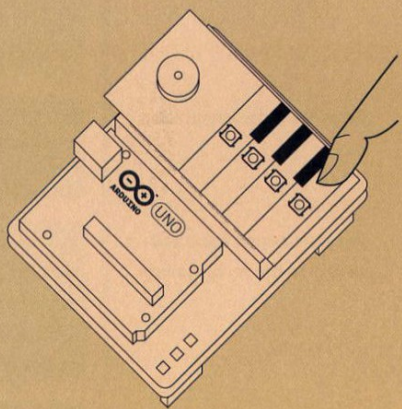
1. Conectar el cable de alimentación y la masa a la placa de pruebas como se hizo en proyectos anteriores. Conectar un pin del zumbador a masa. Conectar el otro pin al terminal 8 (salida digital) de Arduino.

2. Conectar los pulsadores en la placa de prueba tal y como se muestra en esta ilustración. La disposición de las resistencias y los pulsadores conectados a esta entrada analógica se conoce con el nombre de circuito mixto. Conectar el primer pulsador directamente a la alimentación positiva. Conectar el segundo, el tercero y el cuarto pulsador a la alimentación positiva a través de una resistencia de 220 ohmios, 10 kilo ohmios y 1 mega ohmio respectivamente. Conectar todas las salidas de los pulsadores entre sí, en un solo punto de unión. Conectar este punto de unión a masa a través de una resistencia de 10Kilo ohmios, y también conectarla a la entrada analógica 0 de Arduino. Cada pulsador junto con la resistencias se comportan como un divisor de tensión en el momento de que un pulsador se presiona.

Piensa en como mejorar este proyecto añadiendo un dibujo que imite un teclado. Mientras que los sintetizadores analógicos antiguos no tenían un diseño atractivo, este teclado es elegante y además digital. Preparar un pequeño trozo de cartón que se pueda cortar para acomodar los pulsadores y el zumbador. Colocar una etiqueta a cada tecla para saber que nota suena en el momento de pulsarla.



1. Dibuja y corta un trozo de papel para los cuatro pulsadores y el zumbador. Decóralo para que parezca el teclado de un piano



2. Coloca el papel sobre los pulsadores y el zumbador. ¡Disfruta de tu creación!

EL CÓDIGO

La matriz

En este programa, será necesario memorizar las frecuencias de las notas musicales que se van a oír cuando se presione un pulsador. Se van a utilizar las frecuencias medias que se corresponden con la cuarta octava de las notas musicales de un piano, C4, D4, E4 y F4, cuyas frecuencias son 262, 294, 330 y 349 ciclos por segundo o hercios. Para poder hacer todo esto se utiliza una nueva clase de variable llamada matriz.
La matriz es una forma de almacenar diferentes valores que están relacionados unos con otros, como las frecuencias de una escala musical, usando un solo nombre. El uso de una matriz permite acceder a la información que ella almacena de una forma rápida y eficiente. Para declarar una matriz se hace igual que con una variable, pero escribiendo a continuación del nombre corchetes cuadrados: []. Después del signo igual la información se puede colocar de varias formas, entre llaves para definir el valor de las variables o sin ellas si solo se define un valor. En la primera línea de la ventana de la derecha se declara una matriz de seis valores llamada botones (int botones[6];).
En la segunda línea ademas de definir una matriz con una sola variable también se indica el valor que va a contener (int botones[0] = 2;).
Para leer o cambiar los valores de las variables de una matriz, se accede individualmente a cada una de estas variables usando el nombre de la matriz y a continuación el número de posición que ocupa dentro de esta matriz. La posición se refiere al orden que ocupa esta variable cuando la matriz es creada. La primera variable ocupa la posición 0, la segunda la posición 1, la tercera la posición 2 y así sucesivamente. Por ejemplo, en la matriz *int notas[] = {262,294,330,349}*, el valor de la variable notas[2] vale 330.

Crear una matriz para guardar las frecuencias

Configurar una matriz para guardar las cuatro notas musicales usando las frecuencias que se muestran en el primer párrafo de esta hoja. Se crea como una matriz global al declararla antes de la función **setup()**.

Establecer la comunicación serie con el ordenador

Dentro de la función **setup(),** se escribe la instrucción que permite realizar la comunicación con el ordenador.

Leer el valor de la entrada analógica para enviarla al monitor serie

En el **loop()**, declarar una variable para guardar el valor que se obtiene al leer el pin de la entrada analógica A0 de Arduino (int ValorTeclaPulsada). Como cada pulsador tiene en serie una resistencia diferente conectada al positivo de la alimentación, en cada uno de ellos aparecerá una tensión diferente cuando estén presionados, por ejemplo, si se presiona el pulsador que tiene la resistencia de 10K en serie (pulsador de la Nota 3) en la entrada A0 aparecerán 2.4V (para +VCC = 4.8V), por tanto el valor que lee Arduino es 512. Para ver estos valores en la pantalla del ordenador se añade la línea "**Serial.println(ValorTeclaPulsada);**".

Utilizar la instrucción if()..else para determinar que nota deberá de sonar

Usando la instrucción **if()...else**, se puede asignar cada uno de los valores que se obtienen al presionar los pulsadores a un tono diferente, por ejemplo, el pulsador Nota 3 con 330Hz, para un valor de 512. Como las resistencias tienen una tolerancia, al presionar el pulsador de la Nota 3 Arduino puede leer un valor diferente a 512, por tanto se establecen unos márgenes (de 505 a 515) para saber que este pulsador ha sido presionado. Después de montar el circuito si alguno de los pulsadores no funciona usar la información del monitor serie para saber cual es el valor que lee Arduino y así ampliar el margen para este pulsador.

```
int botones[6];
// Crear una matriz para 6 variables de tipo entero

int botones[0] = 2;
// Dar a la primera variable de la matriz el valor de 2
```

```
1  int notas[ ] = {262,294,330,349};
```

```
2  void setup() {
3    Serial.begin(9600);
4  }
```

```
5  void loop() {
6    int ValorTeclaPulsada = analogRead(A0);
7    Serial.println(ValorTeclaPulsada);
```

```
8    if(ValorTeclaPulsada == 1023){
9      tone(8, notas[0]);
10   }
```

Reproducir las notas que se correspondan con el valor analógico de la entrada

Después de la función **if()**, se llama a la función **tone()**. El programa se dirige a la matriz para determinar que frecuencia se debe de reproducir. Si la lectura realizada en la entrada analógica A0 coincide con uno de los valores de las instrucciones if, se le indica a Arduino que reproduzca un tono. Es posible que el circuito produzca ruido, ya que los valores pueden fluctuar un poco mientras se presiona un pulsador. Para corregir esta variación, es una buena idea disponer de un rango de valores para verificar que el valor de la lectura coincide con alguno de los valores dentro de este rango. Se utiliza el operador lógico AND (función de multiplicación), representado por "&&", dentro de múltiples instrucciones **if()...else** para comprobar la condición anterior.

Si presiona el primer pulsador (SW1 - Nota 1), la nota [0] (262 Hz) se escuchará a través del zumbador. Si presiona el segundo pulsador (SW2 – Nota 2) se oirá la nota[1] (294 Hz), y si se presiona el tercer pulsador (SW3 – Nota 3) se oirá la tercera nota[2] (330 Hz). Ahora es cuando se puede ver lo útil y manejable que es usar una matriz.

Parar de toca un tono cuando no se presiona nada

Solo una frecuencia se puede reproducir a la vez, así que si se presionan dos pulsadores al mismo tiempo solo se oirá un sonido, aquel que tenga la resistencia más baja de los dos pulsadores presionados.

Para silenciar el zumbador cuando no se presiona un pulsador se utiliza la función **noTone()**, el cual necesita el número del terminal en donde dejar de reproducir el sonido.

COMO SE UTILIZA

Si los valores de las resistencias del circuito montado son los mismos que los mostrados en este proyecto, se deberá oír un sonido a través del zumbador cuando se presione un pulsador. Si no es así, mirar a través del monitor serie el valor que aparece al presionar un pulsador para verificar que se encuentran dentro de los rangos establecidos en las instrucciones **if()...else.** Si se oye un sonido intermitente, incrementar un poco el rango de valores de la instrucción cuyo valor se encuentra cerca del valor mostrado al presionar el pulsador que produce este sonido intermitente.

Presionar varios pulsadores al mismo tiempo, para ver que tipo de valor aparece en el monitor serie. Usar estos nuevos valores para producir mas sonidos. Experimentar con diferentes frecuencias para aumentar el número de tonos que puede reproducir este teclado. Es posible ver cuales son las frecuencias de las notas musicales en esta página: *arduino.cc/frequencies*

Si se reemplaza el circuito mixto de resistencias por sensores analógicos, ¿será posible usar la información que se obtiene de estos sensores para crear un instrumento musical más dinámico? Se podrían usar estos valores para cambiar la duración de una nota o, como el proyecto del Theremin, crear una gama de sonidos que varían con la iluminación.

```
11   else if(ValorTeclaPulsada >= 990 && ValorTeclaPulsada <= 1010){
12     tone(8, notas[1]);
13     }
14    else if(ValorTeclaPulsada >= 505 && ValorTeclaPulsada <= 515){
15     tone(8, notas[2]);
16     }
17    else if(ValorTeclaPulsada >= 5 && ValorTeclaPulsada <= 10){
18     tone(8, notas[3]);
19     }
```

```
20   else{
21     noTone(8);
22   }
23 }
```

La función **tone()** es útil para generar tonos, pero tiene algunas limitaciones. Solo puede crear formas de onda cuadrada, no ondas con formas senoidales o triangulares. Las ondas cuadradas en absoluto parecen ondas. Como se pudo ver en la figura 1 del proyecto número 6 (página 71), la onda cuadrada esta formada por una serie de pulsos en estado bajo (0V) y en estado alto (+5V).

Ahora es posible formar una banda para usar este teclado, pero habrá que tener en mente algunas cosas: solo un tono se puede reproducir a la vez y la función **tone()** interferirá con la instrucción **analogwrite()** sobre los pins 3 y 11, en caso de que se quiera usar junto con estos pins para aumentar el número de sonidos que este teclado pueda generar.

Las matrices son útiles para guardar el mismo tipo de información junta; se accede a la información de una matriz por los números índice los cuales hacen referencia a las variables individuales que contiene. Los montajes mixtos con resistencias son una forma fácil de obtener un mayor número de entradas analógicas al conectarlos a una entrada analógica dentro de un sistema.

08

PULSADOR

DIODO LED

RESISTENCIA DE 10 KILO OHMIOS

RESISTENCIA DE 220 OHMIOS

INGREDIENTES

RELOJ DE ARENA DIGITAL

EN ESTE PROYECTO, SE VA A MONTAR UN RELOJ DIGITAL DE ARENA QUE ENCIENDE UN DIODO LED CADA 10 SEGUNDOS. DE ESTA MANERA AL USAR UN TEMPORIZADOR CONSTRUIDO EN ARDUINO SE PODRÁ SABER CUANDO TIEMPO SE TRABAJA EN UN PROYECTO

Descubra: datos de tipo largo, creación de un temporizador

Tiempo: **30 MINUTOS**

Nivel: **medio**

Proyectos en los que se basa: **1,2,3,4**

Hasta ahora, cuando se ha querido que suceda algo al pasar un intervalo de tiempo específico con Arduino se ha usado la instrucción delay(), la cual es útil pero un tanto limitada. Cuando se ejecuta delay() Arduino se paraliza hasta que se termine el tiempo especificado dentro de esta instrucción. Esto significa que no es posible trabajar con las señales de entrada y salida mientras está paralizado. Delay tampoco es muy útil para llevar un control del tiempo transcurrido. Resulta un tanto engorroso hacer algo cada 10 segundos utilizando para ello delay junto con este tiempo de retraso.

La función **millis()** ayuda a resolver estos problemas. Realiza un seguimiento del tiempo que Arduino ha estado funcionando en mili segundos. Se ha utiliza esta función en el proyecto número 6 (página 70 Theremin controlado por luz) para establecer un tiempo de 5 segundos que permita calibrar los niveles de iluminación del sensor.

Hasta ahora se han declarado variables del tipo **int**. Una variable **int** (entero) es un número de 16 bit, esto significa que puede guardar números decimales entre -32768 y 32767. Estos valores numéricos pueden parecer muy grandes, pero si Arduino cuenta 1000 veces por segundo usando la función **mills()**, llegará a superar el rango de valores entre -32768 y 32767 (65536 números) en 65.5 segundos. Los datos de tipo largo o **long** pueden guardar números de 32 bits (entre -2147483648 y 2147483647). Ya que no es posible contar el tiempo hacia atrás usando números negativos, la variable para guardar el tiempo que hay que usar en la función **mills()** se llama **unsigned long**. Cuando un tipo de datos es llamado *unsigned* (sin signo), solo trabaja con números positivos. Esto posibilita realizar cuentas mucho mayores. Una variable del tipo unsigned long puede contar más de 4294 millones de números. Son bastantes números para ser contados con **mills()** ya que le llevaría hacerlo más de 50 días. Al poder comparar el tiempo que cuenta **millis()** con un tiempo en concreto, es posible determinar la cantidad de tiempo que ha transcurrido con respecto a ese tiempo.

Cuando se gira el reloj de arena sobre si mismo, el interruptor de inclinación cambia de estado, y comenzará un nuevo ciclo de encendido de los diodos LED.

El interruptor de inclinación trabaja igual que un interruptor normal, pero en esta aplicación se comporta como un sensor de encendido/apagado. Aquí se usará como una entrada digital, ya que proporciona dos niveles lógicos diferentes, “0” y “1”. Los interruptores de inclinación son únicos a la hora de detectar la orientación o inclinación de un objeto. En su interior disponen de una pequeña cavidad con una bola de metal. Cuando el interruptor se gira la bola de metal se mueve en su interior rodando hasta uno de los extremos de la cavidad, haciendo que dos terminales se conecten entre sí de forma que se cierra el circuito que está conectado a la placa de pruebas. En ese momento el reloj de arena digital comenzará a contar un tiempo de 60 segundos encendiendo un LED, de los 6 de que dispone, cada 10 segundos.

MONTANDO EL CIRCUITO

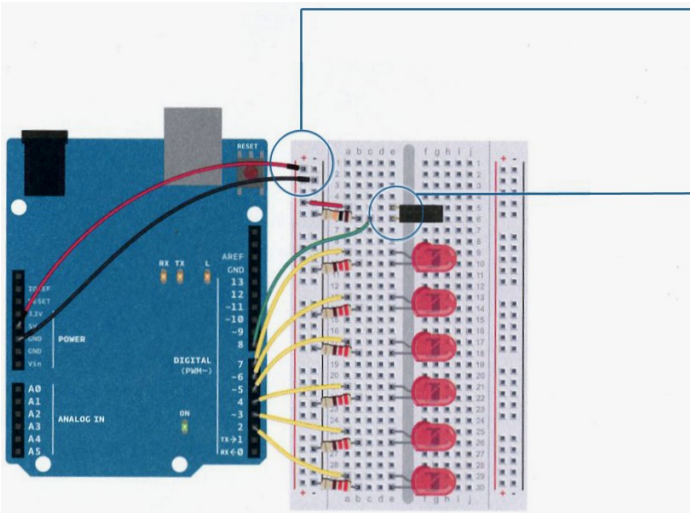


Figura 1

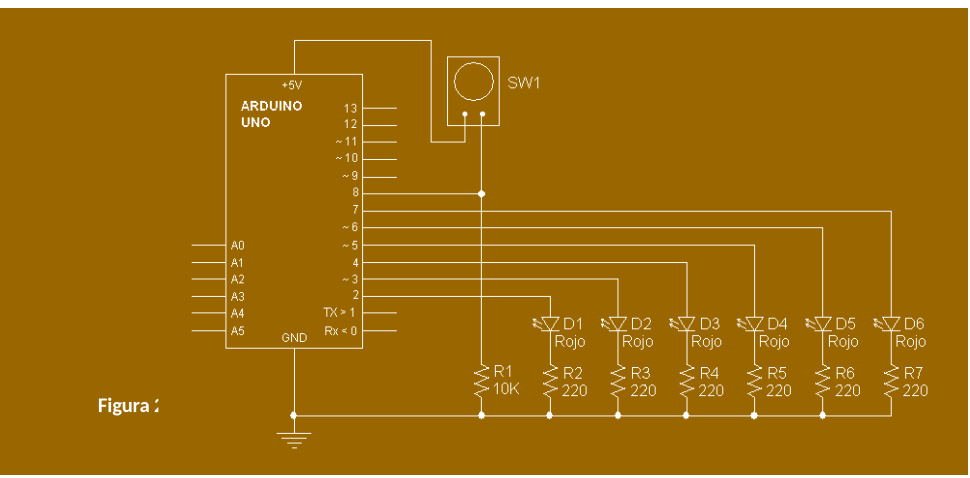


Figura 2

1. Conectar el cable de alimentación positivo y negativo (cables rojo y negro) a la placa de pruebas
2. Conectar el ánodo de cada uno de los seis LEDs a los pins digitales de 2 a 7. Conectar el otro pin de los LED a masa y a través de una resistencia de 220 ohmios .
3. Conectar un pin del interruptor de inclinación al positivo de alimentación +5V de Arduino. Conectar el otro pin a masa usando una resistencia de 10 kilo ohmios. Conectar el punto de unión de esta resistencia con el terminal del interruptor al pin digital número 8.

No es necesario mantener este proyecto conectado al ordenador una vez ha sido cargado el programa en la placa de Arduino. Se podría intentar construir una pequeña caja de cartón para colocar el montaje en su interior junto con la batería. También se puede dibujar en una de las caras de esta caja unos indicadores numéricos para los diodos led, de manera que vaya indicando el tiempo transcurrido después de haber girado la caja.

Los interruptores de inclinación son geniales, componentes de bajo coste para determinar la inclinación de algún objeto. ***Los acelerómetros*** son otro tipo de sensores que proporcionan mucha más información. También son significativamente más caros. Para saber si un objeto se gira cuando se mueve la utilización de un sensor de inclinación es la opción más adecuada.

EL CÓDIGO

Declarar variable de tipo constante

Es necesario definir varias variables del tipo global para conseguir que el programa funcione. Para comenzar, se crea una variable del tipo constante llamada **PinInterruptor**. Este será el nombre del pin del sensor de inclinación que detecta el giro del circuito.

Crear una variable para guardar el tiempo

Crear una variable del tipo **unsigned long**. Se utiliza para almacenar el tiempo en que cambia de estado (apagado-encendido) cualquiera de los seis LEDs.

Definir variables para las salidas y la entrada

Crear una variable que almacene el estado del sensor de inclinación y otra que guarde el estado anterior del mismo. Se usarán estas dos variables para comparar la posición (cerrado o abierto) del interruptor del sensor de un ciclo completo al siguiente.

Crear una variable llamada Led. Se usará para contar qué LED será el próximo en encenderse. Comienza por la salida digital de pin 2 (LED D1).

Crear una variable para establecer el intervalo entre los encendidos

La última variable que se va a crear guarda el intervalo de tiempo de encendido entre cada LED. Será una variable del tipo **long**. Para establecer un tiempo de 10 segundos (el tiempo que pasa entre el encendido de un LED y el siguiente) el valor de esta variable valdrá 10.000 mili segundos. Para aumentar o disminuir este tiempo simplemente hay que cambiar este valor, por ejemplo, para un tiempo de 1 minuto poner 60000.

Establecer como trabajan los pins digitales

Dentro de la función **setup()** se configuran los pins que van a trabajar como salidas para encender los diodos LED, en este caso los pins 2 al 7. Se utiliza la instrucción **for()** para crear un bucle que define en solo tres líneas de código estos seis pins como salidas **OUTPUT**. También aquí dentro se define el pin (8 dentro de PinInterruptor) que se conecta al sensor de inclinación como una entrada **INPUT**.

Averiguar el tiempo que el programa lleva funcionando

Cuando se ejecuta la parte principal de este programa dentro del **loop()**, es necesario saber el tiempo que ha transcurrido desde que Arduino está encendido usando la instrucción **mills()**, y a continuación guardar este dato dentro de una variable local (porque se localiza dentro del bucle) llamada **TiempoActual**.

Evaluar la cantidad de tiempo transcurrido desde el bucle anterior

Utilizando una instrucción **if()**, se verificará si se ha pasado el intervalo establecido para encender un diodo LED. Restando el valor de la variable **TiempoActual** del valor de la variable **TiempoPrevio** se comprueba si este resultado es mayor que el valor de la variable **TiempoIntervalocadaLed**. Si han pasado 10.000 mili segundos (10 segundos), la variable **TiempoPrevio** toma el valor de la variable **TiempoActual**.

```
const int PinInterruptor = 8;
```

```
unsigned long TiempoPrevio = 0;
```

```
int EstadodelInterruptor = 0;
int EstadoPreviodelInterruptor = 0;

int Led = 2;
```

```
long TiempoIntervalocadaLed = 10000;
```

```
void setup() {
  for(int x = 2;x<8;x++){
    pinMode(x, OUTPUT);
  }

  pinMode(PinInterruptor, INPUT);
}
```

```
void loop() {
  unsigned long TiempoActual = millis();
```

```
  if(TiempoActual - TiempoPrevio > TiempoIntervalocadaLed){
    TiempoPrevio = TiempoActual;
```

Encender un LED y prepararse para encender el siguiente

La variable **TiempoPrevio** indica la última vez que un LED fue encendido. Una vez se ha configurado **TiempoPrevio,** se enciende un LED y se incrementa el valor de la variable Led. Una vez que transcurra el tiempo indicado en la variable **TiempoIntervalocadaLed**, se encenderá el siguiente LED.

Comprobar si todos los LEDs están encedidos

Se añade otra instrucción **if()** en el programa para comprobar si el LED conectado al pin número 7 está encendido. En caso de que se cumpla esta condición no se hace nada. Se podrá decidir más adelante que sucede cuando ha transcurrido este tiempo de 1 minuto.

Leer el estado del sensor

Ahora que se ha comprobado el tiempo, también se comprueba si el sensor de inclinación ha cambiado de estado. Se lee el valor del interruptor dentro de la variable **EstadodelInterruptor**.

Poner a cero los valores de las variables si es necesario

Con otra instrucción **if()** se comprueba si el interruptor del sensor de inclinación ha cambiado de posición con respecto a otro estado anterior. La operación != comprueba si la variable **EstadodelInterruptor** no es igual a la variable **EstadoPreviodelInterruptor**. Si son diferentes, se apagan todos los diodos LEDs, y la variable LED toma el valor del primer pin de salida de Arduino (2), además de poner a cero el temporizador de los LEDs al igualar la variable **TiempoPrevio** a **TiempoActual**.

Establecer el estado actual al estado anterior

Al final del bucle **loop()**, se guarda el estado actual del interruptor del sensor de inclinación dentro de la variable **EstadoPreviodelInterruptor**, de esta manera se puede comparar con el valor que se guarde en **EstadodelInterruptor** en la siguiente ejecución del programa o bucle **loop()**.

COMO SE UTILIZA

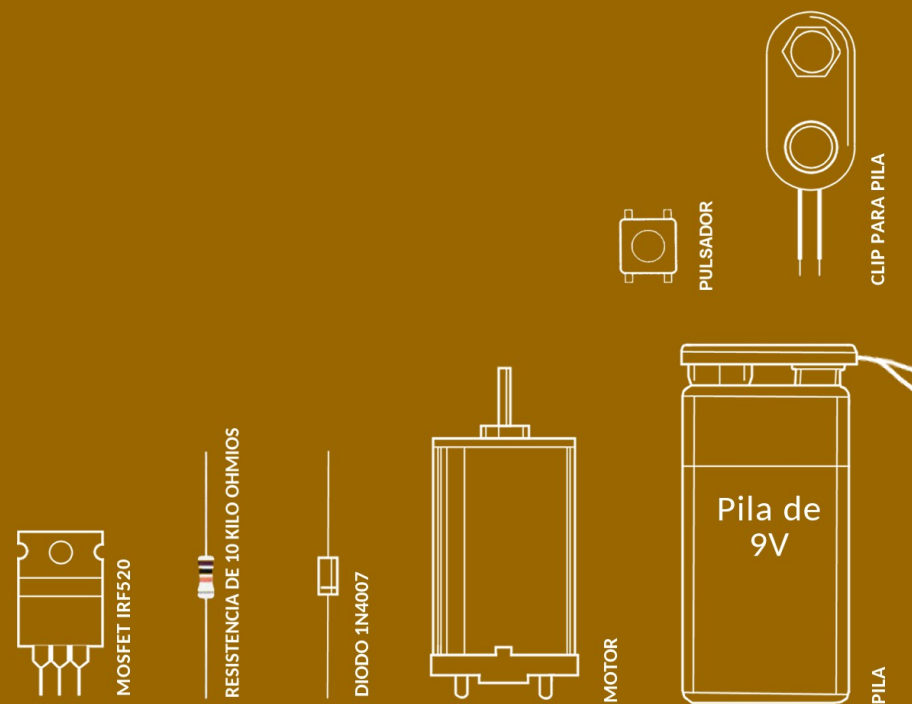
Una vez que se ha programado la tarjeta Arduino, comprobar el tiempo de un minuto usando un reloj. Después de que hayan pasado 10 segundos se debe de encender el primer diodo LED. Cada 10 segundos debe de encenderse un LED. Al final de los 60 segundos todos los diodos LEDs deberán de estar encendidos. Si se mueve el circuito en cualquier momento y se consigue que el sensor de inclinación cambie de estado, todos los diodos LEDs se apagarán y comenzará uno nuevo ciclo de encendido de los LEDs cada 10 segundos.

```
17  digitalWrite(Led, HIGH);
18    Led++;
19    if(Led == 7){
20    }
21   }
22  EstadodelInterruptor = digitalRead(PinInterruptor);
23   if(EstadodelInterruptor != EstadoPreviodelInterruptor){
24    for(int x = 2;x<8;x++){
25      digitalWrite(x, LOW);
26    }
27    Led = 2;
28    TiempoPrevio = TiempoActual;
29   }
30   EstadoPreviodelInterruptor = EstadodelInterruptor;
31  }
```

Cuando el reloj alcanza los 60 segundos se encienden los seis diodos LEDs y permanecen en este estado. ¿Puede pensar como conseguir que el circuito llame la atención cuando se alcance este tiempo? Son buenos indicadores para hacer esto generar un sonido o hacer que los diodos LEDs se pongan en intermitencia. La variable Led se puede comprobar para ver si todos los LEDs están encendidos, ese es un buen lugar para colocar varias instrucciones en el programa para que se llame la atención al finalizar el tiempo. A menos que el reloj se rellene con arena, las luces podrían ir encendiéndose hacia arriba o hacia abajo dependiendo de la orientación del sensor de inclinación. Se puede pensar como usar la variable que indica el estado del interruptor del sensor para indicar la dirección de encendido de los LEDs.

Para medir una cantidad de tiempo entre eventos, se utiliza la función millis(). Los números que genera esta función son más grandes que los que se puede guardar usando el tipo de variable int, por eso es necesario usar el tipo de variable unsigned long para guardar los valores que genera mills()

09



INGREDIENTES

RUEDA DE COLORES MOTORIZADA

CONSEGUIR QUE ARDUINO HAGA GIRAR UNA RUEDA DE COLORES USANDO UN MOTOR

Descubra: transistores, cargas de gran corriente y tensión

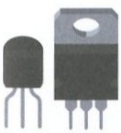
Tiempo: **45 MINUTOS**

Nivel: **medio**

Proyectos en los que se basa: **1,2,3,4**

El controlar motores con Arduino es más complicado que simplemente encender y apagar diodos LEDs por un par de razones. Primero, los motores necesitan más cantidad de corriente que los pins de salida de Arduino pueden suministrar, y segundo, los motores pueden generar su propia corriente a través de un proceso llamado inducción, la cual puede dañar el circuito si no se tiene en cuenta y no se corrige. Sin embargo, los motores hacen posible mover objetos físicos, haciendo de esta forma los proyectos más interesantes. Por tanto, aunque son mas complicados merece la pena el usarlos.

Mover cosas consume gran cantidad de energía. Los motores normalmente necesitan una corriente muy superior a la que Arduino puede suministrar. Además algunos motores también necesitan una mayor tensión para funcionar. El motor para comenzar a funcionar, así como para mover una carga pesada unida a él, consumirá tanta corriente como pueda necesitar. Arduino solo puede suministrar una corriente máxima de 40 mili amperios (40 mA) desde sus terminales digitales, es decir, una corriente mucho menor que la mayoría de los motores necesitan para funcionar.



Los transistores son componentes que permiten controlar grandes cantidades de corriente y altas tensiones a partir de una corriente pequeña (muy inferior a 40mA) de una salida digital de Arduino. Hay muchas clases de transistores, pero todos trabajan bajo el mismo principio. Se puede pensar en los transistores como interruptores digitales, es decir, no se cierran ni se abren usando un dedo, sino mediante el control de un parámetro eléctrico (tensión o corriente). En el los transistores del tipo Mosfet, como el que se utiliza en este proyecto, se aplica una tensión al pin de control del transistor, llamado puerta, de manera que el transistor se cierra entre sus dos terminales extremos llamados fuente y surtidor, tal y como lo haría un interruptor real. Este tipo de transistores se conoce con el nombre de unipolares y siempre funcionan por una tensión aplicada a su terminal de control, esto equivale a usar un dedo cuando se cierra o se abre un interruptor real. Así que de esta manera es posible suministrar una gran corriente y tensión a un motor para encenderlo o apagarlo usando Arduino.

Los motores son un tipo de dispositivo de inducción. La inducción es un proceso por el cual la circulación de una corriente eléctrica a través de un cable (bobina) genera un campo magnético alrededor de este cable. Cuando a un motor se le aplica electricidad, el cable que se enrolla en su interior formando una bobina produce un campo magnético. Este campo hace que el eje (el cilindro que sobresale del motor) comience a dar vueltas.

Al revés también funcionan: un motor puede producir electricidad cuando se hace girar su eje. Intentar unir un diodo LED entre los dos terminales del motor y a continuación girar el eje de este motor a mano. Si no sucede nada, volver a girar el eje pero en sentido contrario a como se hizo antes. El diodo LED debe de alumbrar. Acaba de realizar un pequeño generador con el motor.

Cuando se deja de suministrar energía a un motor, continuará girando, debido a la inercia que posee. Cuando el motor esta girando produce una tensión de sentido contrario a la dirección de la corriente que se le aplica para que funcione. Se ha visto este efecto si ha usado el diodo LED con el motor y ha girado su eje para hacer que alumbre (párrafo anterior). Esta tensión inversa inducida, puede dañar el transistor que lo controla. Por esta razón es necesario colocar un diodo en paralelo con el motor, de manera que la tensión inducida pase a través del diodo y no afecte al transistor. El diodo solo permite el paso de la corriente en una sola dirección, protegiendo de esta forma al resto del circuito.

MONTANDO EL CIRCUITO

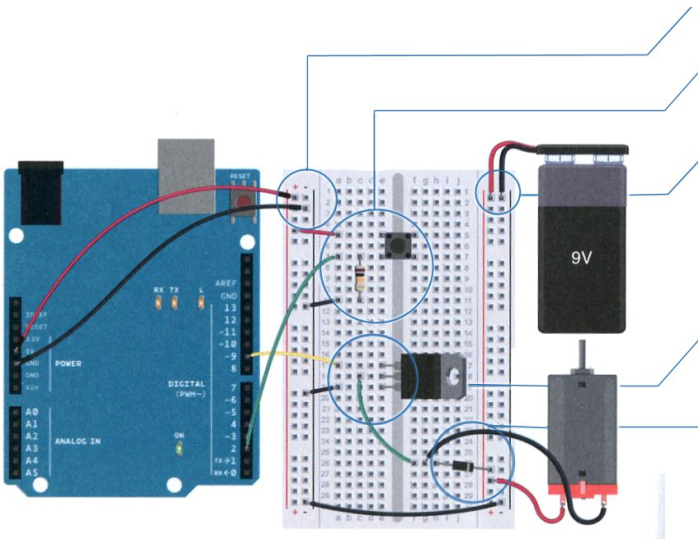


Figura 1

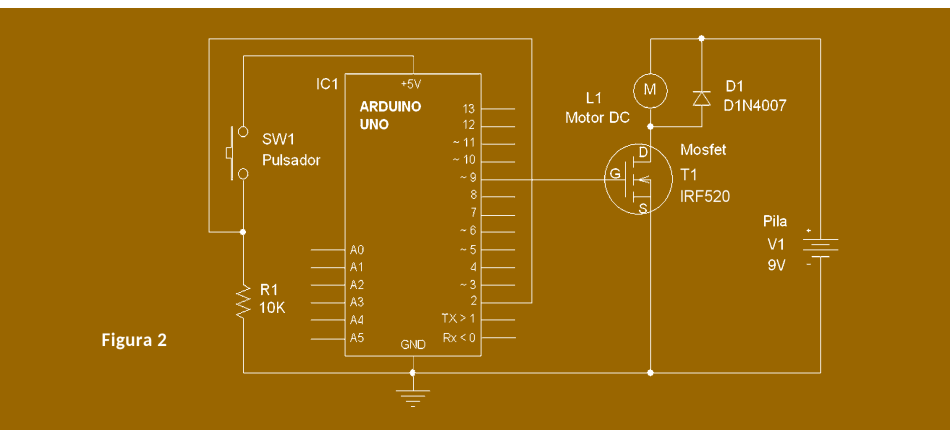


Figura 2

1. Conectar los cables de alimentación positivo y negativo a la placa de pruebas desde la tarjeta Arduino.

2. Añadir un pulsador a la placa de pruebas, conectar uno de sus terminales al positivo de alimentación y el otro terminal al pin digital 2 de Arduino. Añadir una resistencia de 10 kilo ohmios conectada a masa por un extremo y por el otro conectada al punto de unión del pulsador con este pin 2 de Arduino.

3. Cuando se utilizan circuitos alimentados con diferentes tensiones, es necesario conectar las masas de ambos circuitos juntas para así tener una masa común. Conectar la pila de 9V la placa de pruebas usando el clip para este tipo de pilas. Conectar la masa de la batería (cable negro) a la masa de Arduino en la placa de pruebas mediante un cable que hace de puente, como se puede ver en la figura 1. Conectar el cable rojo de la pila a la placa de pruebas en la segunda columna, columna de la derecha, siendo está conexión la que alimenta al motor directamente.

4. Conectar el transistor IRF520 sobre la placa de pruebas de manera que la cara con una mayor superficie metálica quede mirando hacia la derecha (al lado contrario de la placa Arduino). Conectar el pin digital 9 de Arduino al terminal superior del transistor. Este terminal recibe el nombre de **puerta**. Un cambio en la tensión que se aplica al terminal de **puerta** hace que el transistor conecte entres sí sus otros dos terminales (se comporta como un interruptor cerrado). Conectar el cable negro del motor al terminal central del transistor, el cual recibe el nombre de **drenador**. Cuando Arduino activa el transistor al suministrar una tensión al terminal de **puerta,** este terminal (**drenador**) se conectará a un tercer terminal llamado **fuente**, el cual a su vez está conectado a masa. De esta forma al conectar el **drenador** y el **surtidor** entre sí se conecta el cable negro del motor directamente a masa y a través de este transistor, comenzando en ese momento el motor a girar.

5. Lo siguiente, conectar los cables de alimentación del motor a la placa de pruebas, tal y como se puede ver en esta ilustración. El último componente que hay que añadir es el diodo. El diodo es un componente que tiene polaridad, solo se puede montar de una forma determinada en el circuito. Observar que el diodo en uno de sus extremos tiene una franja blanca. Este extremo del diodo es el negativo y se conoce con el nombre de cátodo (por donde sale la corriente). El otro extremo es el positivo o ánodo (por donde entra la corriente). Conectar el ánodo del diodo al cable negro del motor (masa) y el cátodo de este diodo al cable rojo del motor (alimentación positiva), tal y como se ve en esta figura 1. Parece que el diodo está conectado al revés, en realidad sí que es así. El diodo ayuda a eliminar cualquier tensión inversa inducida producida por el motor y que pueda retornar al transistor estropeándolo. Recordar, la tensión inversa inducida siempre aparece con una polaridad contraria a la tensión que se aplica para producirla.

Los LEDs también son diodos, se puede haber asombrado al saber que sus terminales también son llamados ánodo y cátodo. Existen muchas clases de diodos, pero todos ellos tienen algo en común, que solo permiten que la corriente los atraviese en el sentido de ánodo a cátodo y no al revés. Por tanto el diodo en este circuito está conectado al revés con respecto a la tensión de alimentación de 9V pero al derecho con respecto a la tensión inversa inducida del motor.

EL CÓDIGO

Declarar variables de tipo constante y de tipo entero

El código de este programa es muy similar al código usado en el proyecto número 2 de la página 32 (Interface de nave espacial) para encender varios diodos LEDs. Lo primero que hay que hacer es crear algunas constantes para los pins del pulsador y del motor así como crear una variable de tipo entero llamada **EstadodelPulsador** para guardar el estado del pulsador (si está o no presionado).

Declarar las direcciones de los terminales o pins

Dentro del bloque del **setup()**, declarar a través de la instrucción **pinMode()** el pin del motor como salida (**OUTPUT**) y el pin del pulsador como entrada (**INPUT**).

Leer la entrada, poner la salida en alto si el pulsador está presionado

El bloque de ejecución en bucle **loop()** se localiza más adelante. Dentro de este bloque se procede a verificar el estado del pulsador usando la variable **EstadodelPulsador** con la instrucción **digitalRead()**.

Si el pulsador está presionado, se pone en estado alto (HIGH) el terminal del motor a través de la variable de tipo constante PindelMotor, esto hace que en el terminal digital número 9 de Arduino aparezca una tensión de +5V que activa el transistor IRF520 el cual a su vez alimenta al motor y lo hace girar. Si el pulsador no está presionado se coloca un estado bajo (LOW) en este terminal PindelMotor, no llegando una tensión a la puerta del transistor y por lo tanto el motor no funciona.

Los motores tienen unas tensiones óptimas de funcionamiento. Si se aplica solo el 50% de la tensión nominal el motor también va a funcionar. Si se varía la tensión aplicada al motor también se podrá variar la velocidad a la que gira. No conviene variar demasiado esta tensión de funcionamiento del motor para evitar que se queme.

Los motores merecen un consideración especial cuando se van a controlar mediante un microcontrolador como es Arduino. Normalmente los microcontroladores no pueden proporcionar la corriente y/o la tensión que necesita un motor para funcionar. A causa de esto es por lo que hay que usar transistores entre el microcontrolador y el motor. Hay que colocar unos diodos de protección para evitar que se quemen los transistores que controlan directamente la alimentación aplicada a los motores.

```
const int PindelPulsador = 2;
const int PindelMotor = 9;
int EstadodelPulsador = 0;

void setup() {
  pinMode(PindelPulsador, INPUT);
  pinMode(PindelMotor, OUTPUT);
}

void loop() {
  EstadodelPulsador = digitalRead(PindelPulsador);

  if (EstadodelPulsador == HIGH) {
    digitalWrite(PindelMotor, HIGH);
  }
  else {
    digitalWrite(PindelMotor, LOW);
  }
}
```

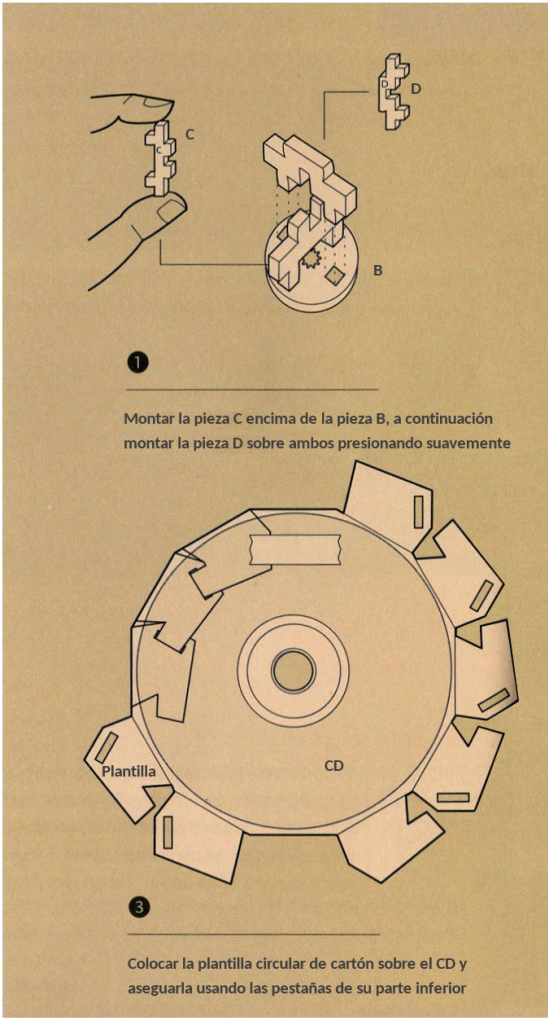
Los transistores son componentes electrónicos de estado sólido, no tienen partes mecánicas que se puedan mover, como en el caso de un relé. Debido a esto los transistores se pueden activar y desactivar (cambiar de estado) a gran velocidad. Por ejemplo, los transistores de conmutación rápida pueden trabajar con tiempos de control del orden de nano segundos. Piensa como conectar un potenciómetro a una entrada analógica para usar un pin digital de salida PWM para controlar la velocidad del motor a través del transistor. ¿Que sucederá con la velocidad del motor si a la vez se varía la tensión que se le aplica? Utilizando los patrones de color que se muestran sobre la plantilla del CD y variando la velocidad del motor ¿se pueden conseguir diferentes efectos visuales?

COMO SE UTILIZA

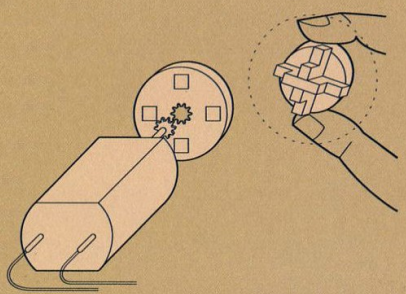
Ensamblar el soporte de unión con el CD como muestra el paso uno de la ilustración inferior, y a continuación unirlo al motor como muestra el paso 2 (parte superior de la ilustración de la página siguiente). Unir la plantilla circular con dibujos de colores que se incluye en este KIT con el CD tal y como muestra el paso 3. Encajar la cruz de madera del soporte del motor al agujero del CD y a continuación usar un poco de pegamento para evitar que se suelte cuando el motor funcione. Ahora para probar el circuito alimentar la placa Arduino a través del puerto USB de un ordenador y usar el clip de pila de 9V con una pila de este tipo. Cuando se presione el pulsador de la placa de pruebas el motor comenzará a girar gran velocidad, por tanto será necesario sujetar con la mano el motor con la plantilla montada para evitar que arrastre al circuito cuando funcione.

MUY IMPORTANTE
(Nota del traductor)

El motor consume del orden de 100 mA sin carga cuando está funcionando, y más de 280 mA cuando tiene montada la plantilla, así que no es nada recomendable usar una pila normal de 9V, ya que se descargará en seguida. La mejor solución es usar una batería de 9V de 1.2 amperios/hora del tipo gel-sodio. En caso de hacerlo no es necesario usar el clip para pila de 9V, dos simples cables conectados a la batería y a la placa de pruebas y en los mismos lugares en donde se colocan los cables del clip.

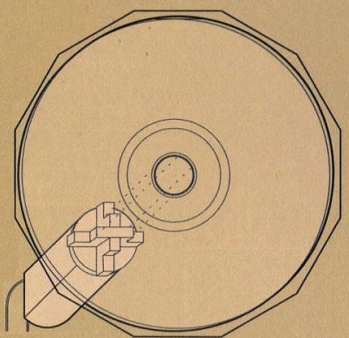


Cuando el motor funcione lo hará a gran velocidad, produciendo una gran fuerza centrífuga. Hay que tener cuidado de que la plantilla o algunas de sus piezas salga volando e impacte sobre el ojo de alguien. También sujetar bien el motor antes de que comience a girar, para evitar que se mueva en el momento de hacerlo. Puede poner en práctica la sugerencia de controlar la velocidad del motor mediante un potenciómetro y también puede cambiar el diseño de la plantilla para conseguir diferentes efectos visuales.



2

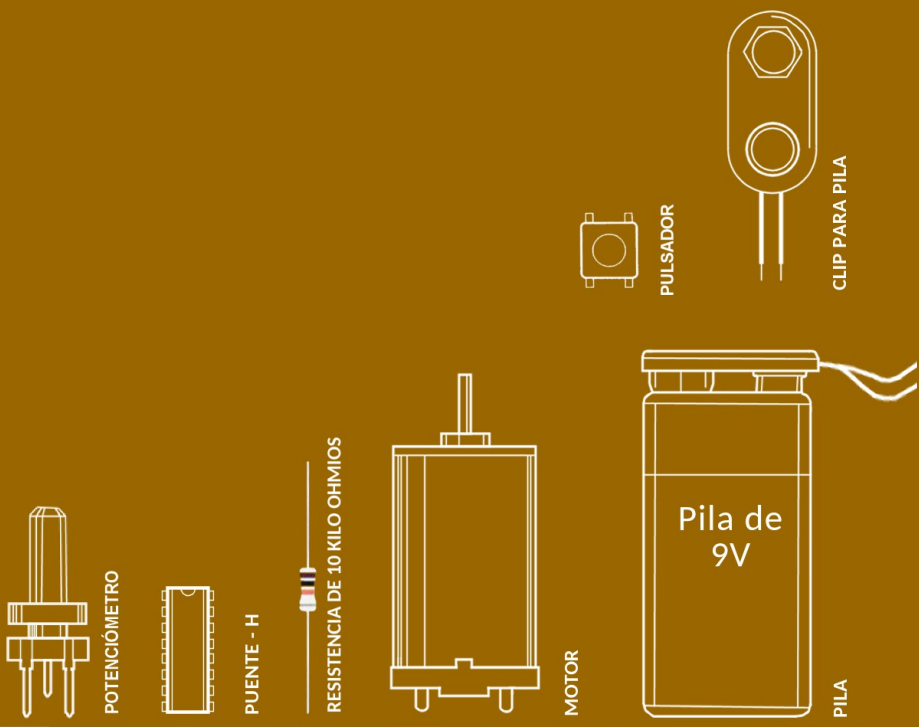
Suavemente presionar el eje del motor dentro del agujero de la parte trasera de la pieza B



4

Unir el CD a la cruz formada por las piezas B y D. Usar un poco de pegamento para evitar que CD se despegue al girar el motor.

10



INGREDIENTES

ZOÓTROPO

CREACIÓN DE IMAGENES EN MOVIMIENTO QUE SE MUEVAN HACIA DELANTE Y HACIA ATRÁS CON ARDUINO CUANDO SE CONECTA A UN MOTOR MEDIANTE UN PUENTE EN H Y USANDO ALGUNAS IMÁGENES FIJAS

Descubra: puentes en H

Tiempo: **30 MINUTOS**

Nivel: **alto**

Proyectos en los que se basa: **1,2,3,4,9**

Antes de que llegase Internet, la televisión e incluso antes de las películas de cine, algunas de las primeras imágenes en movimiento fueron creadas con una herramienta llamada zoótropo. Los zoótropos crean la ilusión de movimiento a partir de un grupo de imágenes fijas las cuales tienen pequeños cambios entre ellas. Se compone de un tambor circular con unos cortes en su superficie. Cuando el tambor gira se mira a través de los cortes y los ojos perciben las imágenes fijas de su interior como si se estuviesen movimiento. La ranuras ayudan a evitar que las imágenes se conviertan en una gran mancha borrosa cuando se mueven, además la velocidad a la cual estás imágenes aparecen crean la sensación de una única imagen que se está moviendo. Originalmente, estos tambores se giraban a mano o con un mecanismo de arranque.

En este proyecto, usted montará su propio zoótropo el cual producirá la animación de una planta carnívora. Se producirá el movimiento usando un motor. Para hacer este sistema aún más avanzado, se añadirá un pulsador para controlar la dirección de giro del motor, además de otro pulsador para encender y apagar el circuito y un potenciómetro para controlar la velocidad del motor.

En el proyecto anterior de la rueda de colores motorizada se consiguió hacer girar el motor en una sola dirección. Si se aplica un cable de alimentación positiva y otro para la masa el motor girará en una dirección, si se invierten estos cables el motor girará en la dirección contraría. Esto no es muy práctico de hacer cada vez que se quiera cambiar el sentido de giro, así que se va a usar un componente llamado puente H para invertir la polaridad de la tensión aplicada al motor y así hacerlo girar en una dirección o en otra.

Los Puentes H son un tipo de componentes conocidos con el nombre de circuitos integrados. Los circuitos integrados o ICs son componentes que tienen en su interior circuitos muy grandes (formado a su vez por cientos o miles de componentes electrónicos) pero que ocupan muy poco espacio dentro de su encapsulado. Estos circuitos integrados ayudan a simplificar los circuitos más complejos al poderlos colocar como componentes individuales que se pueden reemplazar (se suelen montar sobre un soporte o zócalo que permite el cambiarlos con facilidad, como en el caso del microcontrolador de Arduino Uno Rev3). Por ejemplo, el puente H que se va a usar en este proyecto dispone en su interior de un determinado número de transistores en su interior. En caso de que quisiera reproducir el circuito interno de este puente H necesitaría otra placa de pruebas para montar en ella todos los componentes que contiene este circuito integrado.

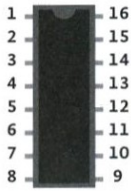


Figura 1

En un circuito integrado se puede acceder a sus circuitos internos a través de los terminales que tienen a ambos lados. Existen diferentes tipos de circuitos integrados con diferente número de terminales, no todos estos terminales se van a usar en cada circuito. Algunas veces es recomendable referirse al número de pin en lugar de referirse a su función. Cuando se mira un circuito integrado, la parte que tiene un muesca indica su parte superior. Se puede identificar el número de terminales al contarlos desde su parte superior izquierda en una U, como se muestra en la figura 1.

MONTANDO EL CIRCUITO

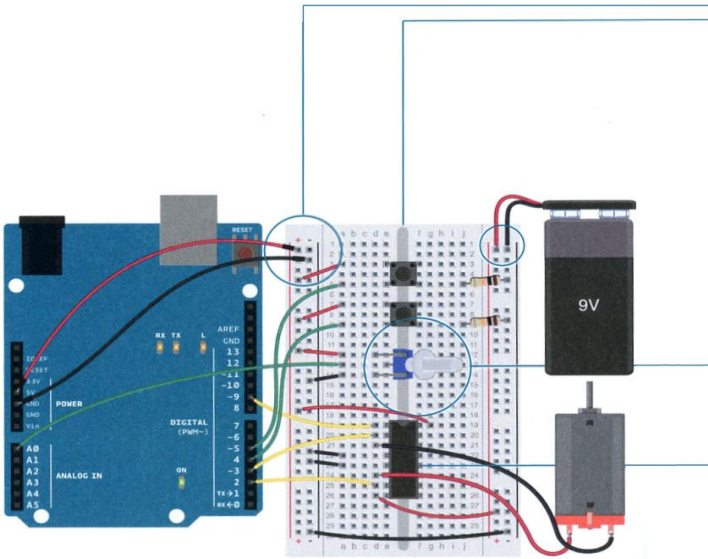


Figura 2

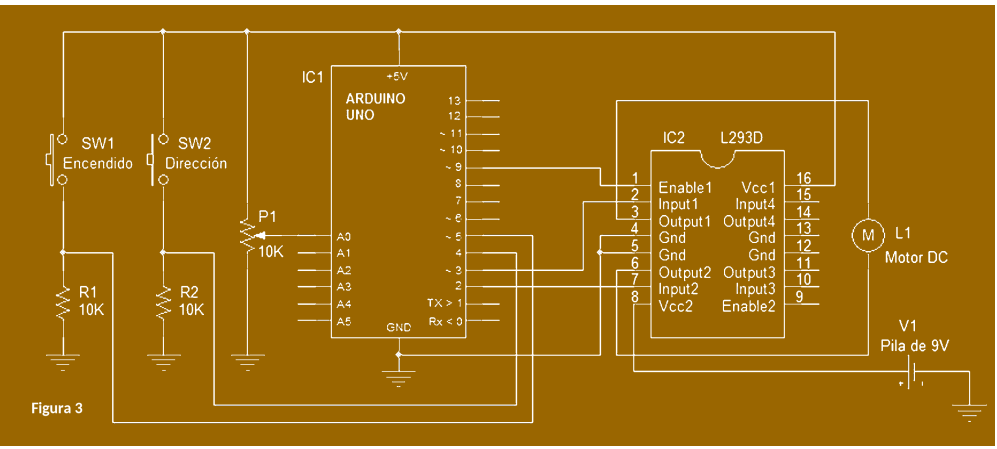


Figura 3

1. Conectar los cables de alimentación positivo y negativo a la placa de pruebas desde la tarjeta Arduino.

2. Añadir dos pulsadores (normalmente abiertos) a la placa de pruebas, conectando un extremo de sus terminales al positivo de la alimentación. Añadir una resistencia de 10 Kilo ohmios en serie con cada resistencia conectando su otro extremo a masa. El punto de unión de uno de estos pulsadores con una resistencia se conecta al terminal digital número 4 de Arduino, se usará como control de dirección, mientras que el otro pulsador se conecta al terminal 5, y se usará para encender y apagar el motor.

3. Conectar el potenciómetro a la placa de pruebas. Conectar uno de sus terminales extremos a +5V y el otro a masa. Unir el terminal central al pin de entrada analógico 0 de Arduino. Este potenciómetro se usará como control de velocidad del motor.

4. Colocar el circuito integrado (puente en H) en el centro de la placa de pruebas, tal y como se puede ver en la figura 2. Conectar el terminal 1 de integrado al terminal número 9 de Arduino. Este es el terminal de activación del integrado. Cuando recibe una tensión de +5V hace que el motor funcione, en cambio cuando no recibe tensión hace que el motor se pare. Se usará este terminal para modular el ancho de pulso de este puente en H y de esta forma variar la velocidad del motor.

5. Conectar el terminal número 2 del integrado al pin digital 3 de Arduino. Conectar el pin 7 al pin digital 2. Estos terminales se usarán para comunicarse con el puente en H, indicando en que dirección deberá de girar el motor. Si el pin 3 está **LOW** y el pin 2 está **HIGH,** el motor girará en una dirección. Si el pin 2 está **LOW** y el pin 3 está **HIGH** el motor girará en la dirección contraria. Si estos dos terminales están en estado **LOW** o en estado **HIGH** a la vez el motor parara de girar.

6. Este circuito integrado recibe su tensión de alimentación de +5V mediante el terminal 16. Los terminales 4 y 5 se conectan los dos a masa.

7. Conectar el motor a los terminales 3 y 6 del integrado. Estos dos terminales encenderán o apagaran el motor dependiendo de las señales que reciba el integrado en los terminales 2 y 7.

8. Colocar el conector para la pila de 9V (sin conectar la pila) a las columnas de positivo y de masa de la placa de pruebas. Conectar la masa de Arduino a la columna del extremo derecho de la placa de pruebas en donde se conecta el cable negro del conector de la pila (masa). Conectar el terminal 8 del integrado (puente en H) a la columna donde se conecta el cable de alimentación positiva de la pila. Este es el terminal a través del cual el integrado alimenta el motor cuando funciona. Asegurarse de que las alimentaciones de +9V y de +5V no están juntas (error de conexión), ya que deben de estar separadas. Solo las masas de ambas alimentaciones deberán de estar conectadas entre sí.

EL CÓDIGO

Nombrar las constantes

Crear constantes para los pins de salida y de entrada

Crear variables para memorizar el estado el programa

Utilizar variables para guardar los valores de las variables de entrada. Se utilizan para detectar el cambio de estado de ambos pulsadores, comparado el estado dentro un ciclo con el siguiente ciclo, igual a lo que se hizo en el proyecto del reloj de arena digital (página 86). De esta forma, además de almacenar el estado actual, será necesario guardar el estado anterior de cada pulsador.

Crear variables para controlar el motor

La variable **DireccionMotor** guarda la dirección de giro del motor, y la variable **ActivarMotor** es la que indica si el motor está girando o no.

Declarar los pins digitales como entradas y salidas

Dentro del bloque del **setup()**, configurar la dirección de cada pin de entrada y salida.

Apagar el motor

Poner la variable **PindeActivacion** en estado **LOW** (bajo) para comenzar, de esta manera el motor no girará.

Leer el estado el pulsador

En la función **loop()**, leer el estado del pulsador de arranque o encendido para guardar su estado dentro de la variable **EstadoPulsadorArranque**.

```
1  const int PindeControl1 = 2;
2  const int PindeControl2 = 3;
3  const int PindeActivacion = 9;
4  const int PinDireccionGiro = 4;
5  const int PinEncendidoApagado = 5;
6  const int PinPotenciometro = A0;
```

```
7  int EstadoPulsadorArranque = 0;
8  int EstadoPrevioPulsadorArranque = 0;
9  int EstadoPulsadorDireccion = 0;
10 int EstadoPrevioPulsadorDireccion = 0;
```

```
11 int ActivarMotor = 0;
12 int VelocidadMotor = 0;
13 int DireccionMotor = 1;
```

```
14 void setup() {
15  pinMode(PinDireccionGiro, INPUT);
16  pinMode(PinEncendidoApagado, INPUT);
17  pinMode(PindeControl1, OUTPUT);
18  pinMode(PindeControl2, OUTPUT);
19  pinMode(PindeActivacion, OUTPUT);
```

```
20  digitalWrite(PindeActivacion, LOW);
21 }
```

```
22 void loop() {
23  EstadoPulsadorArranque =
        digitalRead(PinEncendidoApagado);
24  delay(1);
25  EstadoPulsadorDireccion =
        digitalRead(PinDireccionGiro);
26  VelocidadMotor = analogRead(PinPotenciometro)/4;
```

EL CÓDIGO

Comprobar si los pulsadores han sido presionados

Si existe una diferencia entre el estado actual del pulsador de arranque con respecto a como estaba en el ciclo anterior, y el pulsador se encuentra en estos momentos en HIGH, se pone la variable **ActivarMotor** en el estado contrario al que se encuentra actualmente, si esta en “1” se pone en “0” y viceversa.
Se leen los valores de las posiciones de los pulsadores y del potenciómetro. Se almacenan estos valores en sus respectivas variables

Verificar si la dirección ha cambiado

Verificar si el valor de la variable del pulsador de dirección es actualmente el mismo que en el ciclo anterior. Si es diferente, cambiar la variable de dirección del motor. Solo existen dos formas de que el motor pueda girar, así que esta variable alternará entre dos estados para hacer que el motor gire en un sentido o en el sentido contrario. La forma de lograr esto es utilizando el operador de inversión: **DireccionMotor = !DireccionMotor**

Cambiar los pins para hacer funcionar el motor en una determinada dirección

La variable **DireccionMotor** determina en que dirección gira el motor. Para establecer una dirección de giro, se configuran los pins de control uno en **HIGH** y el otro en **LOW** (gira en un sentido). Si se pone de nuevo uno en **LOW** y el otro en **HIGH** girará en sentido contrario a como lo hizo antes. Cuando la variable **DireccionMotor** cambia, invierte el valor de los pins de control

Si el pulsador de dirección ha sido presionado, se conseguirá que el motor gire en dirección contrario al invertirse el estado de las variables **PindeControl**.

Variar la velocidad del motor si esta funcionando

Si la variable **ActivarMotor** esta a 1, se establece la velocidad de giro del motor usando la instrucción **analogwrite()** para regular el ancho de pulso (PWM) de la tensión de activación del motor en el terminal 9 de Arduino (variable **PindeActivacion**). Si **ActivarMotor** esta a 0, entonces el motor se para al poner su valor a 0 con la instrucción **analogWrite** (todo el ancho de pulso en estado bajo).

Guardar los estados actuales de los pulsadores para el siguiente ciclo

Antes de salir de un ciclo loop(), se guarda el estado actual de los pulsadores como un estado previo de las variables para la siguiente vez que se ejecute el programa.

```
27  if(EstadoPulsadorArranque != EstadoPrevioPulsadorArranque){
28    if(EstadoPulsadorArranque == HIGH){
29      ActivarMotor = !ActivarMotor;
30      }
31    }
```

```
32  if(EstadoPulsadorDireccion != EstadoPrevioPulsadorDireccion) {

33    if (EstadoPulsadorDireccion == HIGH) {
34      DireccionMotor = !DireccionMotor;
35    }
36  }
```

```
37  if(DireccionMotor == 1) {
38    digitalWrite(PindeControl1, HIGH);
39    digitalWrite(PindeControl2, LOW);
40  }

41  else {
42   digitalWrite(PindeControl1, LOW);
43   digitalWrite(PindeControl2, HIGH);
44  }
```

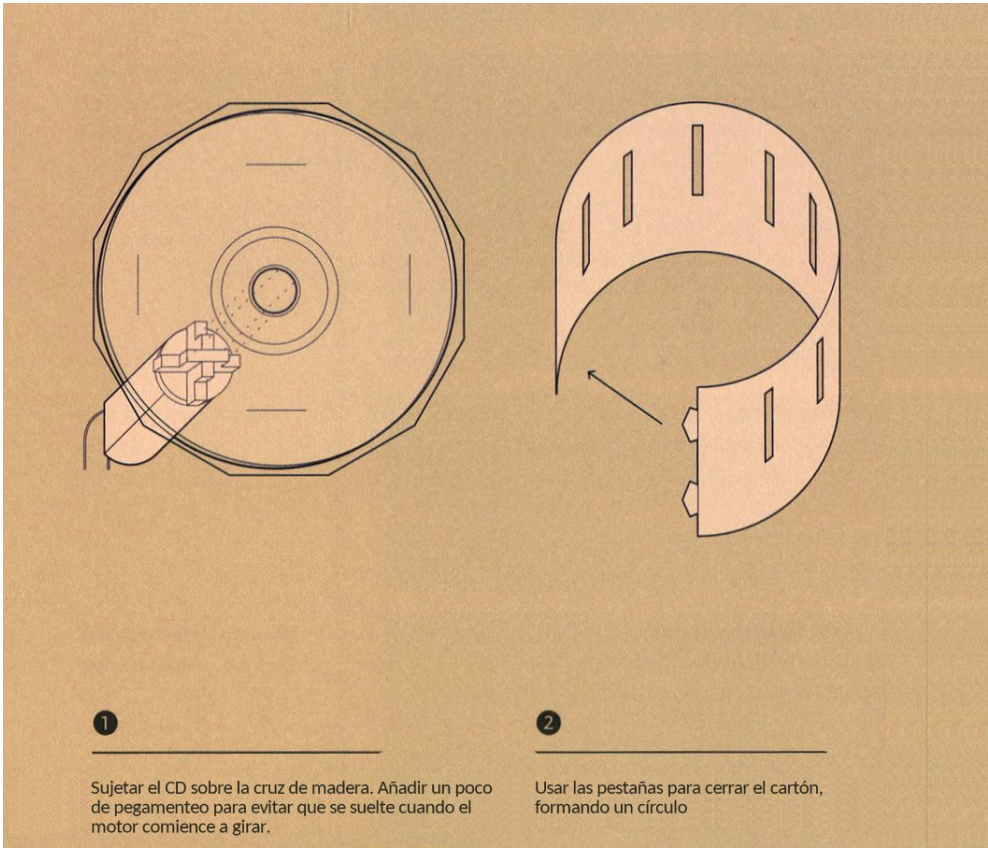
```
45  if(ActivarMotor == 1){
46    analogWrite(PindeActivacion, VelocidadMotor);
47  }
48  else {
49    analogWrite(PindeActivacion, 0);
50  }
```

```
51  EstadoPrevioPulsadorDireccion =
        EstadoPulsadorDireccion;
52  EstadoPrevioPulsadorArranque = EstadoPulsadorArranque;
53 }
```

COMO SE UTILIZA

Una vez que se comprueba que el circuito funciona, desconectar la pila y el USB del circuito

Conectar la placa Arduino al ordenador. Enchufar la pila de 9V al clip de plástico. Cuando se presione el pulsador de arranque el motor deberá de comenzar a girar. Si se gira el mando del potenciómetro la velocidad del giro del motor deberá aumentar o disminuir. Presionando el pulsador de arranque otra vez el motor se parará. Presionar el pulsador de dirección para comprobar que el motor cambia el sentido de giro. También si se gira el mando del potenciómetro, se podrá ver como aumenta o disminuye gradualmente la velocidad de giro del motor dependiendo del giro de este mando así como del sentido en que se hace.



1

Sujetar el CD sobre la cruz de madera. Añadir un poco de pegamenteo para evitar que se suelte cuando el motor comience a girar.

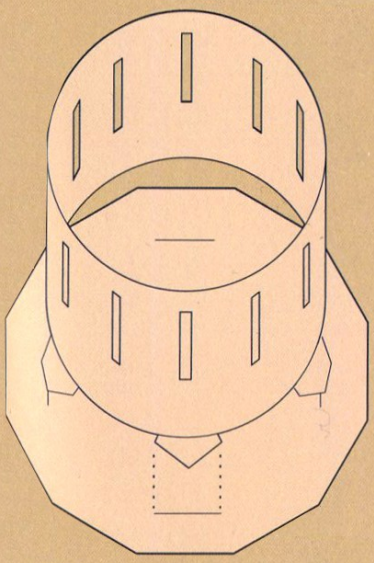
2

Usar las pestañas para cerrar el cartón, formando un círculo

A fin de construir el zoótropo, es necesario disponer de la rueda de colores que se uso en el proyecto número 9 (página 94), así como de una plantilla de color negro que incluye las ranuras a través de las cuales hay que mirar. Se sujeta esta plantilla sobre la rueda de colores a través de unas pestañas. Una vez que el CD esta firmemente conectado al eje del motor, con la plantilla negra encima y las ranuras abiertas, colocar el motor en vertical, de manera que se pueda mirar a través de las ranuras (asegurarse de que el CD no esté demasiado cerca del motor. Se debe de ver mirando a través de las ranuras como una secuencia de imágenes fijas “se mueven” cuando el motor gira. Si van demasiado rápidas o lentas girar el mando del potenciómetro para ajustar la velocidad de la animación.

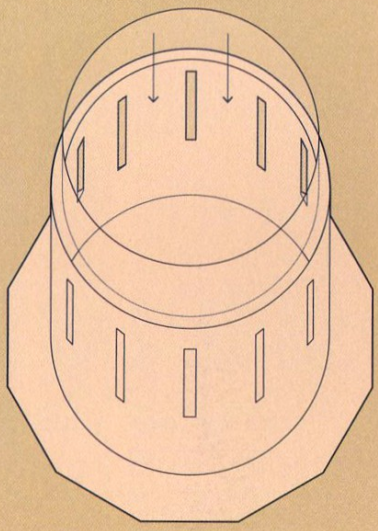
Presionar el pulsador de dirección para ver como la animación se reproduce en sentido contrario a como lo estaba haciendo. El zoótropo y las imágenes proporcionadas en este kit son el punto de partida: puede intentar experimentar con nuevas animaciones, utilizando la plantilla de esta animación como una referencia.

Para hacer esto, comenzar con una imagen básica. Identificar un punto en esta imagen y realizar pequeños cambios en el resto de los fotogramas. Intentar volver a la imagen original para conseguir que la animación se pueda reproducir continuamente sin saltos.



3

Insertar las cuatro pestañas en la base de la plantilla sobre el CD .



4

Insertar la tira de papel con las imágenes dentro del zoótropo.

El zoótropo trabaja gracias a un fenómeno llamado “persistencia de la visión”, algunas veces abreviado con el término POV (Persistence Of Vision). POV describe la ilusión del movimiento que se crea cuando nuestros ojos observan imágenes fijas con pequeñas variaciones entre ellas y en rápida sucesión. Si efectuá una búsqueda online para el texto “POV display”, encontrará muchos proyectos realizados por personas que aprovechan este efecto, a menudo con diodos LEDs y Arduino.

Fabricar una base para sujetar el motor. Una pequeña caja de cartón con un agujero podría ser un buen remedio, de esta forma podrá dejar sus manos libres para manejar los pulsadores y el potenciómetro. Además, al hacerlo de esta forma, le será mucho más fácil enseñar este trabajo a todo el mundo.

Puede añadirle una nueva funcionalidad al proyecto, para conseguir que el zoótropo trabaje también en situaciones de poca luz. Conectar un diodo LED y una resistencia en serie a uno de los pins digitales de salida de Arduino que no se utilice. También añadir un segundo potenciómetro y conectarlo a una entrada analógica. Colocar el diodo LED de manera que pueda alumbrar sobre las imágenes. Usar la entrada analógica para variar el parpadeo del LED de manera que alumbre en el momento que se mire a través de las ranuras de la plantilla. Puede llevar algún tiempo el conseguirlo, pero merece la pena por que el efecto final es realmente espectacular.

11

PULSADOR

RESISTENCIA DE 10 KILO OHMIOS

RESISTENCIA DE 220 OHMIOS

POTENCIÓMETRO

PANTALLA LCD

INGREDIENTES

BOLA DE CRISTAL

CREAR UNA BOLA DE CRISTAL PARA QUE LE HABLE DE SU FUTURO

Descubra: *Pantallas LCD, instrucciones swith/case, random()*

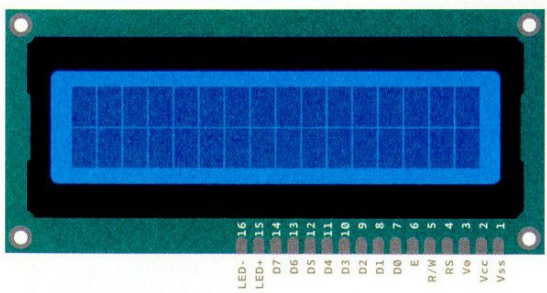
Tiempo: **1HORA**

Nivel: **alto**

Proyectos en los que se basa: **1,2,3**

La bola de cristal le puede ayudar a "adivinar" el futuro. Le hace una pregunta a la bola que todo lo sabe y a continuación deberá de mover la para obtener la respuesta. Las respuestas estarán guardas con anterioridad, pero podrá redactarlas como más le guste. Se usará Arduino para escoger una respuesta de un total de 8 respuestas guardadas. El sensor de inclinación que se incluye en el kit imita el movimiento de la bola cuando se frota para obtener las respuestas.

La pantalla LCD se puede usar para mostrar caracteres alfanuméricos. El de este kit dispone de 16 columnas y de 2 filas, para un total de 32 caracteres. Su montaje sobre la placa de circuito impreso incluye un gran número de conexiones. Estos terminales se utilizan para la alimentación y comunicación, además de indicar lo que tiene que escribir sobre la pantalla, pero no es necesario conectar todos sus terminales, algunos se dejan al aire. Ver la figura número 3 (esquema eléctrico) para ver los terminales que se conectan al circuito.



Todos los terminales de la pantalla LCD. No se conectan todos.

Figura 1

MONTANDO EL CIRCUITO

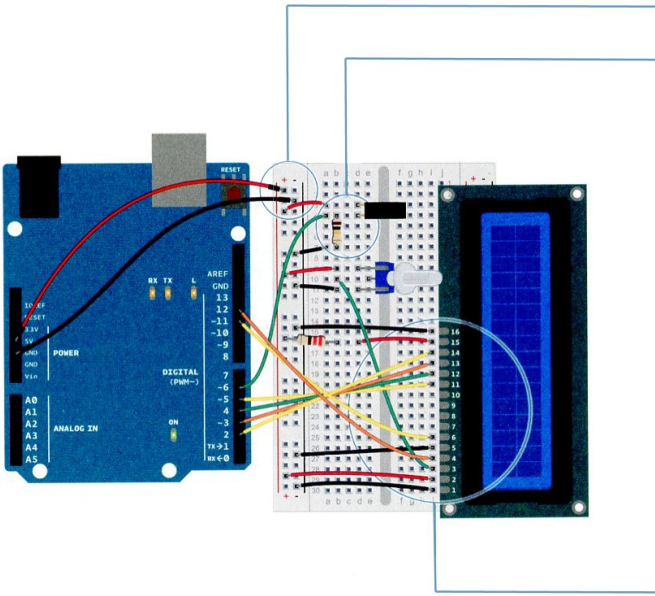


Figura 2

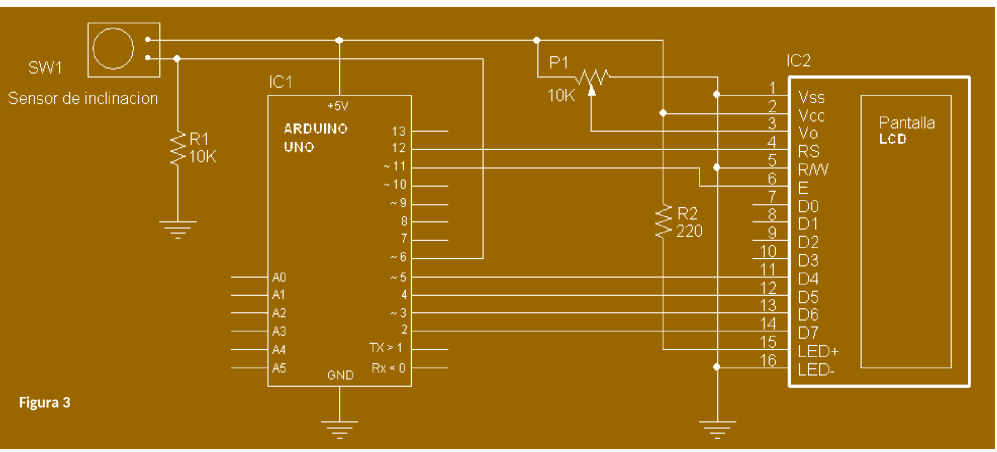


Figura 3

En este esquema la disposición de los terminales de la pantalla LCD no coinciden con el orden físico en la figura 2 (montaje sobre la placa de pruebas). En un esquema, los terminales se reorganizan de una manera lógica de forma que el esquema se pueda leer con una mayor claridad. Esto puede causar una pequeña confusión a los principiantes hasta que realicen el montaje del proyecto.

El circuito no es complicado de montar, pero existen un montón de cables que conectar. Prestar atención cuando se realicen todas las conexiones con los cables, asegurarse que todo está bien conectado.

1. Conectar los cables de alimentación y masa tal y como se muestra en esta ilustración, en un extremo de la placa de pruebas.

2. Colocar el sensor de inclinación en la placa de pruebas y conectar uno de sus terminales a +5V. Conectar el otro terminal a masa y a través de una resistencia de 10K ohmios, el punto de central de unión de esta resistencia y el terminal del sensor se unen al pin número 6 de la placa Arduino. Se configurará como una entrada digital, tal y como se ha realizado en proyectos anteriores.

3. El terminal de selección de registro (**RS**) de la pantalla LCD controla cuando los caracteres aparecerán sobre la pantalla. El terminal de lectura/escritura (read/write) (**R/W**) coloca la pantalla en modo de lectura o de escritura. En este proyecto se usará el modo de escritura. El terminal de activación (enable) (**EN)** le dice a la pantalla que va a recibir una instrucción. Los terminales de datos **(D0-D7)** se utilizan para enviar caracteres de datos a la pantalla. Solo se usarán 4 **(D4-D7)** de estos 8 terminales. Finalmente, la pantalla dispone de un terminal a través del cual se puede ajustar el contraste de la misma (**V0**). Se usará un potenciómetro para realizar este control del contraste.

4. La librería de la pantalla LCD que se incluye con el software de Arduino maneja toda la información de estos terminales y simplifica el proceso de escribir software para mostrar los caracteres. Los dos terminales externos del LCD (**Vss** y LED-) hay que conectarlos a masa. También se conecta a masa el terminal R/W, el cual coloca al LCD en modo de escritura. El terminal de alimentación del LCD (**Vcc)** se conecta directamente a +5V. El terminal LED+ de la pantalla se conecta a través de una resistencia de 220V también a +5V.

5. Conectar: El terminal digital 2 de Arduino al terminal **D7** del LCD, el terminal digital 3 al **D6** del LCD, el terminal digital 4 al **D5** del LCD y el terminal digital 5 al **D4** del LCD. Todos son terminales de datos que le dicen a la pantalla que carácter debe de mostrar.

6. Conectar el terminal **EN** (activación) de la pantalla al terminal 11 de Arduino. El terminal **RS** de la pantalla se conecta al terminal 12 de Arduino. Este terminal habilita la escritura del LCD.

7. Colocar el potenciómetro en la placa de pruebas, conectando uno de sus terminales extremos a la alimentación de 5V y el otro a masa. El terminal central debe de conectarse al terminal **V0** de la pantalla LCD. Este potenciómetro permitirá cambiar el contraste de la pantalla.

EL CÓDIGO

Configurar la librería LiquidCrystal

Primero, es necesario importar la librería **LiquidCrystal**. A continuación, se inicializa esta librería, de la misma forma que se hizo con la librería del servomotor en el proyecto número 5 (indicador del estado de ánimo), indicando en el programa que terminales se van a usar para la comunicación con Arduino.

Una vez que se ha configurado la librería, es el momento de crear algunas variables y constantes. Crear una constante para guardar el número del pin de Arduino al que se conecta el sensor de inclinación (**PindelSensor**), una variable que almacene el estado actual del de sensor (**EstadodelSensor**), otra variable para guardar el estado previo del sensor (**EstadoPreviodelSensor**), y una más (**Contestar)** para elegir la respuesta que se mostrará en la pantalla LCD.

Definir el tamaño de la pantalla

Configurar el pin de Arduino al que se conecta el sensor de inclinación como una entrada con la instrucción **pinMode()** dentro de la función **setup()**. Iniciar la librería LCD e indicarle el tamaño de la pantalla (16 columnas por 2 filas).

Mover el cursor y escribir el primer texto

Ahora es el momento de escribir un pequeño texto de introducción a modo de pantalla de bienvenida antes las 8 respuestas posibles. La función **print()** escribe en la pantalla LCD. Se va a escribir el texto "Preguntame" en la línea superior de la pantalla. El cursor se coloca automáticamente al comienzo de la línea superior y en la primera columna.

Con el fin de escribir en la siguiente línea, hay que indicarle a la pantalla donde se debe de colocar el cursor (setCursor(0,1)). Las coordenadas de la primera columna y de la segunda línea son 0,1 (recuerda que en los ordenadores las coordenadas 0,0 se refiere a la primera columna de la primera fila). Usar la función **lcd.setCursor()** para mover el cursor a la primera columna de la segunda fila y a continuación decirle que escriba la frase "Bola de Cristal".

Ahora, cuando comience el programa, aparecerá en la pantalla LCD "Preguntame Bola de Cristal".
Dentro de la función **loop(),** primeramente se verifica el estado del sensor de inclinación, y a continuación se coloca el valor leído dentro de la variable EstadodelSensor.

Escoger una respuesta aleatoria

Usar la instrucción **if()** para determinar si el sensor está en una posición diferente a como estaba en un estado previo. Si es diferente a como estaba antes y además en estos momentos está en estado LOW (bajo), entonces es el momento de generar una respuesta aleatoria. La función **random()** devuelve un número dentro del rango que se indica dentro del argumento. Para comenzar, el programa guarda 8 posibles respuestas para la bola de cristal. Siempre que la instrucción random(8) es ejecutada, se obtendrá cualquier número comprendido entre 0 y 7. Este número se guarda dentro de la variable Contestar.

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

const int PindelSensor = 6;
int EstadodelSensor = 0;
int EstadoPreviodelSensor = 0;
int Contestar;

void setup() {
 lcd.begin(16, 2);
 pinMode(PindelSensor,INPUT);

 lcd.print("Preguntame");

 lcd.setCursor(0, 1);
 lcd.print("Bola de Cristal");
}

void loop() {
 EstadodelSensor = digitalRead(PindelSensor);

 if(EstadodelSensor != EstadoPreviodelSensor) {
  if(EstadodelSensor == LOW) {
   Contestar = random(8);
```

LCD library reference
arduino.cc/lclibrary

Random reference
arduino.cc/random

Limpiar la pantalla con la función **lcd.clear()**. Estas instrucciones también mueven el cursor a la posición inicial 0,0; la primera columna de la primera fila del LCD. Escribir el texto "**La bola dice**" en pantalla y mover el cursor a la línea siguiente.

Predecir el futuro

La instrucción **switch()** ejecuta diferentes partes del código dependiendo del valor que tenga esta instrucción. Cada una de estas partes del código se llama mediante una instrucción **case switch()** la cual verifica el valor de la variable Contestar (valor como argumento de la instrucción); de esta manera cualquier valor que se genera dentro de la variable Contestar determina cual de las instrucciones case (de 1 a 7)será ejecutada.

Dentro de las instrucciones case, el código será el mismo, pero los mensajes que muestre en pantalla serán diferentes. Por ejemplo, en case 0 el código hace que a través de la función **lcd.print("Si")** aparezca el texto de "**Si**" en la pantalla. Después de la función **lcd.print()** hay otro comando: **break**, que le dice a Arduino donde se localiza el final de la última instrucción case. Cuando se ejecuta **break**, el programa salta al final de la última sentencia de la instrucción switch, la cual siempre será }. Para comenzar se van a crear un total de 8 instrucciones case. Cuatro de las respuestas serán positivas, dos serán negativas, una respuesta que no tiene ni idea y otra respuesta que vuelva a preguntar de nuevo.

Lo último que hay que hacer al final del programa, dentro de la función **loop()**, es asignar el valor de la variable **EstadodelSensor** a la variable **EstadoPreviodelSensor**. Esto posibilita que se pueda realizar un seguimiento del cambio de estado (de 0 a 1 ó de 1 a 0) en el sensor de inclinación la próxima vez que el programa se ejecute.

```
19          lcd.clear();
20          lcd.setCursor(0, 0);
21          lcd.print("La bola dice:");
22          lcd.setCursor(0, 1);

23          switch(Contestar){
24           case 0:
25           lcd.print("Si");
26           break;
27           case 1:
28           lcd.print("Es probable");
29           break;
30           case 2:
31           lcd.print("Ciertamente");
32           break;
33           case 3:
34           lcd.print("Buenas perspectivas");
35           break;
36           case 4:
37           lcd.print("No es seguro");
38           break;
39           case 5:
40           lcd.print("Pregunta de nuevo");
41           break;
42           case 6:
43           lcd.print("Ni idea");
44           break;
45           case 7:
46           lcd.print("No");
47           break;
48            }
49           }
50          }

51  EstadoPreviodelSensor = EstadodelSensor;
52  }
```

Switch Case reference
arduino.cc/switchcase

COMO SE UTILIZA

Para usar la bola de cristal primero hay que darle alimentación a la placa Arduino. Comprobar si en la pantalla del LCD aparece el texto "**Preguntame**" y en la segunda línea "**Bola de Cristal**". Si no puede ver bien los textos deberá ajustar el potenciómetro para variar el contraste hasta que los caracteres se vean correctamente.

Realizar una pregunta a la bola de cristal y a continuación mover hacia arriba y hacia abajo el circuito para que el sensor de inclinación pueda detectar estas variaciones. De esta manera obtendrá una respuesta a su pregunta. Si la respuesta no le satisface, puede preguntar de nuevo.

Intente añadir sus propias respuestas a las instrucciones **print()**, pero teniendo en mente que solo es posible usar 16 caracteres por línea. Asegurarse cuando se añaden nuevas instrucciones case switch, que se ha ajustado el número de opciones que la variable Contestar va a almacenar; recordar que esto se define mediante la instrucción random(8), en este caso 8 respuestas. Si se añaden dos nuevas respuestas habrá que aumentar el número 8 a 10 y añadir dos nuevas instrucciones case switch.

La pantalla LCD trabaja cambiando las propiedades eléctricas de un líquido que se encuentra montado entre cristales polarizados. El cristal solo permite que un tipo de luz pase a través de él. Cuando el liquido entre los cristales esta cargado se vuelve semi sólido de manera que puede representar caracteres al bloquear la luz que incide sobre él.

Las funciones descritas aquí para cambiar el texto que aparece en la pantalla LCD son muy simples. Una vez que tenga claro como funciona puede estudiar la librería de este LCD para descubrir nuevas funciones. Puede intentar que el texto se mueva a lo largo de la pantalla o que se actualice continuamente. Para saber más sobre como trabajar con la librería de la pantalla LCD visitar: arduino.cc/lcd

Una pantalla LCD permite mostrar textos usando la librería LiquidCrystal. Con las instrucciones switch...case controla el flujo de partes del programa al poder comparar una variable con determinados valores.

12

PULSADOR

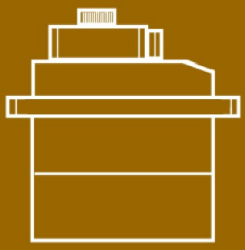
DIODO LED

RESISTENCIA DE 10 KILO OHMIOS

RESISTENCIA DE 220 OHMIOS

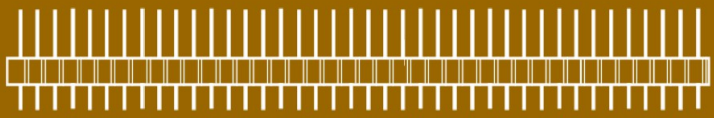
RESISTENCIA DE 1MEGA OHMIO

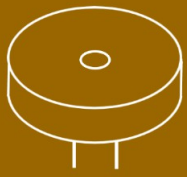
CONDENSADOR DE 100uF



SERVO MOTOR

TIRA DE PINS MACHO (SOLO SON NECESARIOS 3 PINS)





ZUMBADOR PIEZOELÉCTRICO

INGREDIENTES

MECANISMO DE BLOQUEO SECRETO

CONSTRUIR UN MECANISMO DE BLOQUEO SECRETO PARA EVITAR QUE ALGUIEN PUEDA SUSTRAER LAS COSAS QUE TENGA GUARDADAS EN UNA CAJA

Descubra: entrada con un zumbador, escribir sus propias funciones

Tiempo: **1HORA**

Nivel: **alto**

Proyectos en los que se basa: **1,2,3,4,5**

El zumbador usado para reproducir sonidos en los proyectos del theremin y del teclado también se puede usar como un dispositivo de entrada o sensor. Cuando se le aplica una tensión de 5V, este sensor puede detectar las vibraciones las cuales pueden ser leídas por las entradas analógicas de Arduino. Es necesario conectar una resistencia de un valor alto (del orden de 1Mega ohmio) entre la salida y masa para que funcione correctamente, es decir, se realiza el montaje del zumbador en serie con una resistencia de 1Mega ohmio conectada a masa, de manera que el conjunto forma un divisor de tensión.

Cuando un zumbador piezoeléctrico es presionado contra una superficie plana puede vibrar, como por ejemplo sobre una mesa de madera, siendo Arduino capaz de detectar la intensidad de esta vibración. Usando esta información puede comprobar si el número de vibraciones detectadas se encuentran dentro del rango de trabajo establecido. En el código puede contabilizar el número de vibraciones y ver si coinciden con el número de vibraciones almacenadas.

Un pulsador permite bloquear el motor en una posición. Algunos diodos LEDs aportan información: un LED rojo indicará que la caja está bloqueada, un LED verde indicará que la caja no está bloqueada, y un LED amarillo permite saber si se ha recibido el número correcto de vibraciones.

También escribirá su propia función la cual le permitirá saber si una vibración es demasiado fuerte o demasiado débil. El escribir sus propias funciones le ayuda a ahorrar tiempo de programación al llamar siempre a una misma parte del código sin necesidad de volverlo a escribirlo cada vez que se use. Las funciones pueden usar argumentos y devolver valores a través de estos argumentos. En este caso, utilizará una función para determinar el nivel de volumen de la vibración. Si se encuentra dentro del rango correcto se incrementará el valor de una variable.

Es posible construir el circuito simplemente por hacerlo, con ninguna finalidad, pero es mucho más interesante y didáctico hacerlo con un objetivo, en este caso realizar una caja de seguridad en donde guardar algo. Si dispone de una caja de madera o una caja de cartón puede realizar una serie de agujeros en su interior, usar un servomotor para abrir y cerrar un pestillo, y así evitar que cualquiera pueda abrir la caja para coger lo que hay en su interior.

MONTANDO EL CIRCUITO

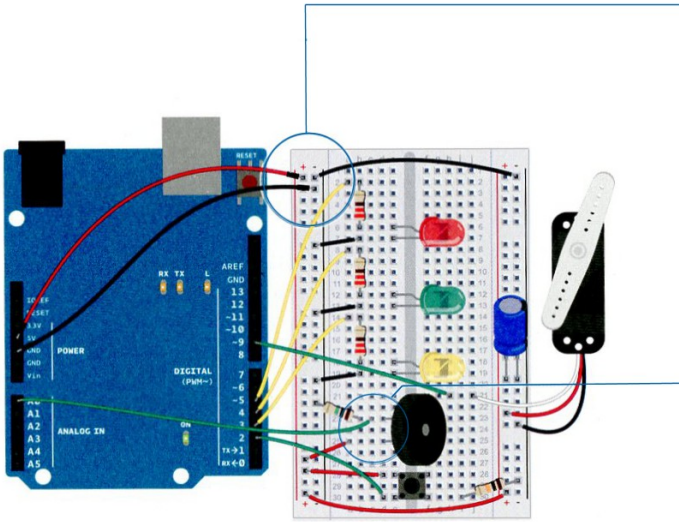


Figura 1

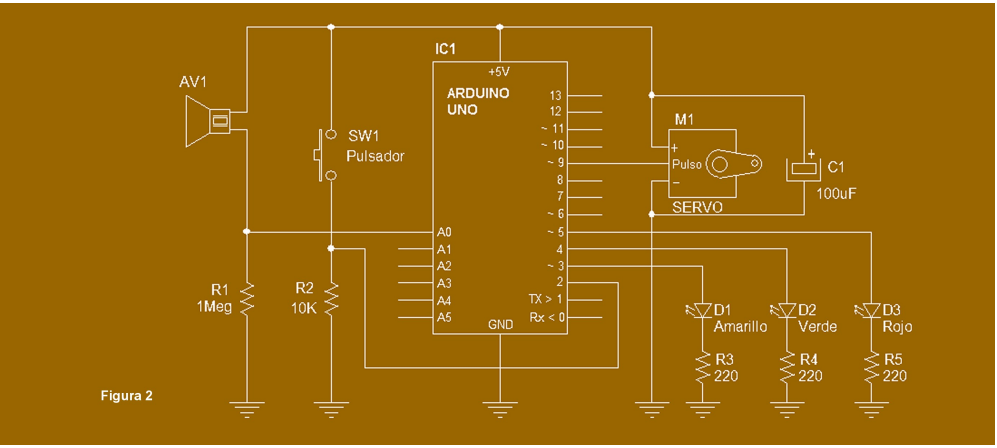


Figura 2

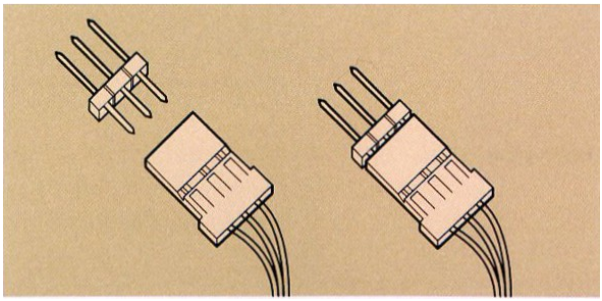
Hay muchas conexiones en la placa, asegurarse que todas estas conexiones están realizadas correctamente.

1. Conectar los cables de alimentación y masa a ambos lados de la placa de pruebas. Colocar el pulsador sobre la placa de pruebas conectando uno de sus terminales a +5V. El otro extremo del pulsador conectarlo a masa a través de una resistencia de 10Kilo ohmios. Conectar la unión de la resistencia y el terminal del pulsador al pin digital número 2 de Arduino

2. Colocar el zumbador sobre la placa de pruebas y a continuación conectar un cable desde uno de sus terminales al positivo de la alimentación. Si el zumbador tiene un cable de color rojo o el símbolo "+", este será el terminal que se conecte al positivo. Si en el zumbador no se indica la polaridad se podrá conectar de cualquier forma. Conectar el otro terminal del zumbador al terminal Analog 0 de Arduino. Colocar una resistencia de 1Mega ohmio entre masa y este terminal conectado a Arduino. Valores de resistencia más bajos harán que el zumbador sea menos sensible a las vibraciones.

3. Cablear los LEDs, conectando los cátodos (patilla corta) a masa, y colocar un resistencia de 220 ohmios en serie con los ánodos. A través de sus respectivas resistencias, conectar el LED amarillo al pin digital 3 de Arduino, el LED verde al pin digital 4, y el LED rojo al pin digital 5.

4. Insertar el conector macho de tres terminales dentro del conector hembra del servomotor (ver figura 3). Conectar el cable rojo a la alimentación y el cable negro a masa. Colocar un condensador electrolítico de 100 micro faradios en las líneas de alimentación de la placa de pruebas para suavizar las variaciones bruscas de corriente y evitar variaciones de tensión, además asegurarse que el condensador se ha conectado con la polaridad correcta. Conectar el terminal de datos del servomotor, el cable blanco, al terminal 9 de Arduino.



El servomotor viene con un conector hembra, así que será necesario añadirle un conector macho con tres terminales para poder conectarlo a la placa de pruebas

Figura 3

EL CÓDIGO

Librería Servo

Como en el proyecto del "Indicador de estado de ánimo", es necesario importar la librería **Servo** y crear una copia para usar el motor.

Constantes útiles

Crear constantes para nombrar las entradas y las salidas

Variables para guardar los valores del pulsador y del zumbador

Crear variables para guardar los valores del pulsador y del zumbador

Umbrales de los golpes

Configurar algunas variables de tipo constante para usar como límites para los niveles máximo y mínimo de los golpes

Variables para el estado del bloqueo y el número de golpes

La variable bloqueado permite saber si el bloqueo está activado o no. Una variable de tipo **boolean** contiene datos que solo pueden ser verdadero – true (1), o falso – false (0). Cuando se aplica tensión al circuito debe de empezar con el mecanismo desbloqueado. La última variable de tipo global guardará el número de golpes válidos que haya recibido.

Configurar la dirección de los pines digitales e inicialización del servo y del puerto serie

Dentro de la función **setup()**, conectar el servo al pin 9.
Establecer los pines de los LEDs como salidas y los pines del pulsador como entradas.

Desbloquear

Inicializar la comunicación serie con el ordenador para poder ver el nivel de los golpes, en que estado se encuentra el bloqueo y el número de golpes válidos que se han efectuado. Encender el diodo LED verde, mover el servo a la posición de desbloqueo, y mostrar el estado actual en el monitor serie indicando que el circuito está en la posición de desbloqueo.

Comprobar el pulsador

Dentro de **loop()**, primero se comprueba si la caja está bloqueada o no lo está. Esto determinará lo que ocurra en el resto del programa. Si está bloqueado se procede a leer el estado del pulsador.

```
#include <Servo.h>
Servo miServo;

const int zumbador = A0;
const int PinPulsador = 2;
const int LedAmarillo = 3;
const int LedVerde = 4;
const int LedRojo = 5;

int ValorGolpe;
int ValorPulsador;

const int GolpesSuaves = 10;
const int GolpesFuertes = 100;

boolean bloqueado = false;
int NumeroGolpes = 0;



void setup() {
  miServo.attach(9);
  pinMode(LedAmarillo, OUTPUT);
  pinMode(LedRojo, OUTPUT);
  pinMode(LedVerde, OUTPUT);
  pinMode(PinPulsador, INPUT);
  Serial.begin(9600);

  digitalWrite(LedVerde, HIGH);
  miServo.write(0);
  Serial.println("La caja esta desbloqueada!");
}



void loop() {
  if(bloqueado == false){
    ValorPulsador = digitalRead(PinPulsador);
```

Bloqueado

Si el pulsador está cerrado (está presionado), cambiar el valor de la variable bloqueado a true (verdadero), indicando que el bloqueo está activado. Apagar el diodo LED verde y encender el LED rojo. Es necesario activar el monitor serie para que aparezca la indicación "La caja está bloqueada!" y de esta manera ver el estado del bloqueo. Se mueve el servo a la posición de bloqueo. Se añade un tiempo de 1 segundo para que el bloqueo tenga tiempo de realizarse.

Leer los golpes en el sensor

Si la variable bloqueo es true, y el bloqueo está activado, leer el valor de la vibración del zumbador y almacenarlo en **ValorGolpe.**

Contar solo los golpes válidos

Las siguientes instrucciones comprueban si se han efectuado menos de tres golpes válidos y si se percibe alguna vibración en el sensor (zumbador). Si ambas condiciones son ciertas, comprueba si el golpe en este momento es válido o no lo es, e incrementa la variable **NumeroGolpes** en caso de ser válido. En esta parte del código es donde se realiza la llamada a la función personalizada **VerificarGolpes()**. Se escribe esta función al final de la función **loop()**. **VerificarGolpes()** comprueba que el valor de la variable **ValorGolpe** es un golpe válido o no lo es, además esta variable forma parte del argumento de esta función. Después de que se ejecute esta función se indica, a través del monitor serie, el número de golpes válidos que faltan para realizar el desbloqueo.

Desbloquear

Comprobar si se han efectuado tres o más golpes válidos. Si esta condición se cumple, cambiar la variable de bloqueo a false (falso), y mover el servomotor a la posición de desbloqueo. Esperar 20 mili segundos para que le de tiempo a mover el servomotor, y cambiar el estado del LED verde a encendido y apagar el LED rojo. Mostrar un mensaje en el monitor serie indicando que "La caja está desbloqueada!".

Cerrar la instrucción **else** y la función **loop()** con un parte de llaves.

Definir una función para comprobar la valided de los golpes

Ahora es el momento de escribir la función personalizada **VerificarGolpes()**. Cuando se escriben funciones personalizadas es necesario indicar si va a devolver un valor o no. Si no va a devolver un valor la función se declara del tipo void, igual que las funciones **loop()** o **setup()**. Si va a devolver un valor se debe de declarar de que tipo (**int, long, float**, etc). En este caso se comprueba si un golpe es válido (true) o no lo es (false) por eso se declara la función de tipo boolean.

```
28      if(ValorPulsador == HIGH){
29          bloqueado = true;
30          digitalWrite(LedVerde, LOW);
31          digitalWrite(LedRojo, HIGH);
32          miServo.write(90);
33          Serial.println("La caja esta bloqueada!");
34          delay(1000);
35         }
36       }
```

```
37       if(bloqueado == true){
38         ValorGolpe = analogRead(zumbador);
```

```
39         if(NumeroGolpes < 3 && ValorGolpe > 0){
40          if(VerificarGolpes(ValorGolpe) == true){
41            NumeroGolpes++;
42          }
43          Serial.print(3-NumeroGolpes);
44          Serial.println(" golpes para abrir");
45         }
```

```
46      if(NumeroGolpes >= 3){
47         bloqueado = false;
48         miServo.write(0);
49         delay(20);
50         digitalWrite(LedVerde, HIGH);
51         digitalWrite(LedRojo, LOW);
52         Serial.println("La caja esta desbloqueada!");
53         NumeroGolpes = 0;
54         }
55       }
56      }
```

```
57      boolean VerificarGolpes(int valor){
```

La misión de esta función será controlar un número (que contiene la variable **ValorGolpe**) para ver si es válida o no. Para pasar el valor de esta variable dentro de la función se crea una variable de tipo entero (**int valor)** dentro del argumento en el momento de crear dicha función.

Comprobar la valided del golpe

En esta función, cada vez se refiere a la variable **Valor** se usará cualquier número que contenga la variable **ValorGolpe** dentro del programa principal y que se transmite a través del argumento de la función. En esta línea del programa se comprueba que el dato que contiene la variable **Valor** es mayor que el dato que contiene la variable **GolpesSuaves** y menor que el dato que contiene la variable **GolpesFuertes**. Por ejemplo, si el dato de **ValorGolpe** vale 25, se pasa a la función a través del argumento con lo cual la variable **Valor** tendrá este dato y a continuación se comprueba que es mayor que 10 y menor de 100, según se puede ver en la línea adjunta de la página siguiente.

Indicando que el golpe es válido

Si el dato de la variable **Valor** se encuentra entre los dos valores anteriores (entre 10 y 100) entonces existe un golpe válido, por tanto el diodo LED amarillo se enciende durante 50 mili segundos y aparece un texto en el monitor serie indicando que se ha producido un golpe válido “Golpe válido de valor ” así como dicho valor.

Función retorno válido

Para permitir al programa principal usar el resultado de la comparación que se acaba de realizar dentro de la función se utiliza el comando **return** (retorno). Este comando **return** también hace que la función finalice cuando es usado: una vez que se ejecuta la función, retorna al programa principal.

Indicando golpe no válido. Función retorno no válido

Si el dato de la variable **Valor** está por debajo del dato de la variable **GolpesSuaves** (10) o por encima de **GolpesFuertes**(10) aparece el siguiente texto en el monitor serie “El valor de golpe no es válido” y se retorna al programa principal a través de return false.

Cerrar la función personalizado usando más de una llave.

COMO SE UTILIZA

Cuando se conecta Arduino a un ordenador y se carga el programa en él, a continuación hay que abrir el monitor serie. Debe de ver como el diodo LED verde se enciende y el servomotor se mueve a la posición de desbloqueo.

En la ventana del monitor serie aparece la frase "La caja esta desbloqueada!".

Ahora para probar el circuito presionar el pulsador. El diodo LED verde se apaga y se enciende el LED rojo y se muestra la frase "La caja esta bloqueada!" en el monitor serie.

Deberá de determinar el nivel de los golpes que consiguen desbloquear el mecanismo efectuando varios golpes sobre el zumbador con diferentes niveles de intensidad. En el momento que el diodo LED amarillo se encienda durante menos de 1 segundo y aparezca la frase "Golpe valido de valor x" sabrá la intensidad de los golpes para desbloquear el mecanismo.

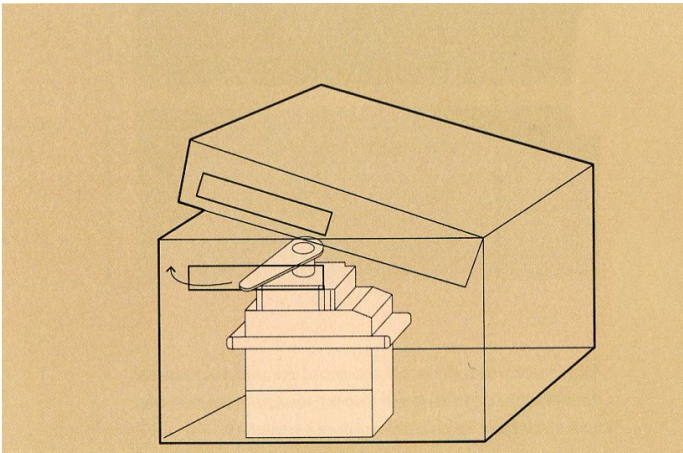
```
58    if(valor > GolpesSuaves && valor < GolpesFuertes){
59      digitalWrite(LedAmarillo, HIGH);
60      delay(50);
61      digitalWrite(LedAmarillo, LOW);
62      Serial.print("Golpe valido de valor ");
63      Serial.println(valor);
64      return true;
65      }
66    else {
67      Serial.print("El valor del golpe no es valido ");
68      Serial.println(valor);
69      return false;
70      }
71  }
```

También en el monitor serie se muestra el valor del golpe válido así como el número de golpes que es necesario efectuar para producir el desbloqueo.

El número de golpes válidos siempre serán 3 y cuando se consigan efectuar estos 3 golpes sobre el zumbador el diodo LED rojo se apagará, el LED verde se encenderá y el servomotor se moverá 90 grados a la posición de desbloqueo, además de aparecer una notificación en el monitor serie indicando que el bloqueo está desactivado.

El valor para el golpe ideal puede variar con respecto a los que aparecen en este ejemplo. Esto depende de un número de variables diferentes, como el tipo de superficie sobre la que se apoya el sensor, si el zumbador está sólidamente sujeto o se mueve cada vez que se golpea, etc. Usando el monitor serie y golpeando varias veces el zumbador se podrá determinar el valor más apropiado de la intensidad del golpe cada vez que se encienda el diodo led amarillo. También se puede hacer usando uno de los ejemplos que incluye el IDE de Arduino y que se localiza dentro "**Ejemplos"**, "**Analog**" y el programa "**AnalogInOutSerial**". Puede acceder a la siguiente página web para ver en más detalle como usar este programa: ***arduino.cc/analogtoserial***

Si se coloca este proyecto dentro de una caja, será necesario hacerle unos agujeros para los LEDs y el pulsador. También será necesario colocar le una palanca al servomotor que permita bloquear la tapa de la caja. Es necesario hacer otro agujero para pasar el cable USB que se conecta al ordenador y así poder ver la intensidad de los golpes que permiten abrirla.
También será necesario reorganizar la colocación de la placa de pruebas y de la tarjeta Arduino, o soldar los LEDS y el pulsador mediante unos cables de forma que se puedan montar en la caja. El soldar consiste en unir dos o más metales, normalmente un cable y un terminal de un componente, usando otro material que permite fijarlos permanentemente y que no permita que se suelten a pesar de que se produzca cualquier movimiento brusco. Sin nunca ha soldado antes, es mejor preguntarle a alguien que ya tenga experiencia en este tipo de trabajos para que pueda ayudarle, o intentar practicar con algunos cables que tenga a mano antes de hacerlo con algunos de los componentes de este proyecto. Cuando se suelda algo esto significa que la conexión realizada al soldar es permanente, así que asegurarse después de realizar la conexión de los LEDs y el pulsador que los cables no se sueltan con facilidad, simplemente puede tirar de ellos para ver si la soldadura está bien realizada. Dirigirse a la página ***arduino.cc/soldering*** para ver una buena explicación de cómo soldar.



❶

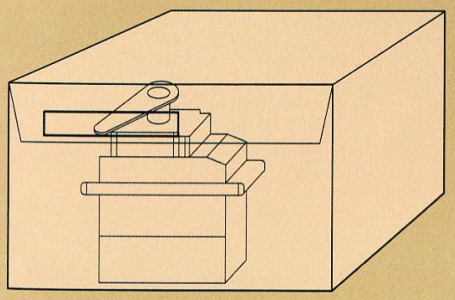
Hacer dos agujeros en la caja: uno en una de las paredes de la base y el otro en un lateral de la tapa. Colocar el servo en el interior de manera que la pestaña se pueda mover del interior al exterior de los agujeros cuando se bloquea.

Escribir sus propias funciones no solo permite controlar el flujo del código con más facilidad, también ayuda a que los proyectos sean más legibles aunque el código se vaya haciendo cada vez más y más grande. Con el tiempo, a medida que escriba más código, podrá disponer de un gran número de funciones que pueda usar en diferentes proyectos, haciendo este proceso más rápido y único por su estilo de programar. (Nota del traductor) Debe de gustarle la programación sino será imposible el hacerlo.

En este proyecto simplemente se cuenta el número de golpes válidos (3) para desbloquear el mecanismo, no importa el tiempo que se tarda en conseguirlo. Se puede mejorar el proyecto al usar un temporizador con la instrucción **mills()**. Usar el monitor serie para comprobar si los golpes válidos se efectúan en un periodo de tiempo específico. Puede volver a mirar el Proyecto del Reloj de Arena Digital (página 86) para ver como trabaja esta instrucción. No está limitado a encontrar los golpes válidos en un rango de tiempo. Puede buscar patrones complejos de golpes sobre la base de la cantidad de vibración y tiempo juntos. Existe un gran número de ejemplos en Internet que tratan de como hacer esto, buscar "**Arduino knock lock**" para encontrar más ejemplos de este tipo de proyecto.

En la página https://www.arduino.cc/en/Tutorial/KnockSensor puede obtener más información de como se usar el zumbador como un sensor de golpes

Los zumbadores de tipo piezo eléctrico se pueden usar como entradas cuando se conectan formando un divisor de tensión junto con una resistencia de gran valor. El diseño de una función es una forma fácil de escribir código que se puede usar para tareas concretas y/o en otros proyectos diferentes.



❷

Fijar el servo en el interior con un poco de pegamento, asegurandose de nuevo que la pestaña del servo puede girar facilmente a través de las ranuras

13

DIODO LED

RESISTENCIA DE 220 OHMIOS

RESISTENCIA DE 1MEGA OHMIO

LÁMINA DE METAL

INGREDIENTES

LÁMPARA SENSIBLE AL TACTO

EN ESTE PROYECTO SE CONSTRUYE UNA LÁMPARA QUE ENCIENDE UNA LUZ Y LA APAGA CUANDO TOCA UNA LÁMINA DE METAL

Descubra: *instalación de bibliotecas de terceros, creación de un sensor táctil*

Tiempo: **45MINUTOS**

Nivel: **medio-alto**

Proyectos en los que se basa: **1,2,5**

Se va a utilizar la biblioteca CapacitiveSensor de Paul Badger en este proyecto. Esta biblioteca le permitirá medir la capacidad de su cuerpo.

La capacidad es una medida que indica la cantidad de carga eléctrica que un cuerpo puede almacenar. La biblioteca comprueba dos terminales de Arduino (uno es un emisor, el otro es un receptor), y mide el tiempo que tardan en alcanzar el mismo estado. Estos terminales se conectarán a un objeto metálico como pueda ser una lámina de aluminio. A medida que se acerque al objeto metálico, su cuerpo absorberá alguna carga eléctrica, haciendo que transcurra más tiempo para que los dos terminales tengan el mismo estado.

Preparando la librería

La versión mas reciente de la biblioteca CapacitiveSensor está aquí: ***arduino.cc/capacitive***. Descargar este fichero en el ordenador y descomprimirlo. Abrir la carpeta de las bibliotecas de Arduino (por defecto se localiza dentro de la carpeta de ***C:\Archivos de programa\Arduino\libraries***). Dentro de esta carpeta crear una carpeta llamada CapacitiveSensor y copiar dentro de ella los archivos que se han descomprimido (localizados dentro de la carpeta arduino-libraries-CapacitiveSensor-7684dff) además de cerrar el IDE de Arduino si está abierto.

Abrir el IDE de Arduino y dentro del menú “Archivos” seleccionar “Ejemplos”, al hacerlo se podrá ver en la parte inferior de la ventana que aparece una nueva entrada con el nombre “CapacitiveSensor”. La biblioteca que se ha añadido contiene un proyecto de ejemplo. Abrir el ejemplo CapacitiveSensorSketch y compilarlo. Sino se produce ningún error sabrá que la biblioteca la ha instalado correctamente.

Para más información sobre las bibliotecas:

arduino.cc/en/Reference/Libraries

MONTANDO EL CIRCUITO

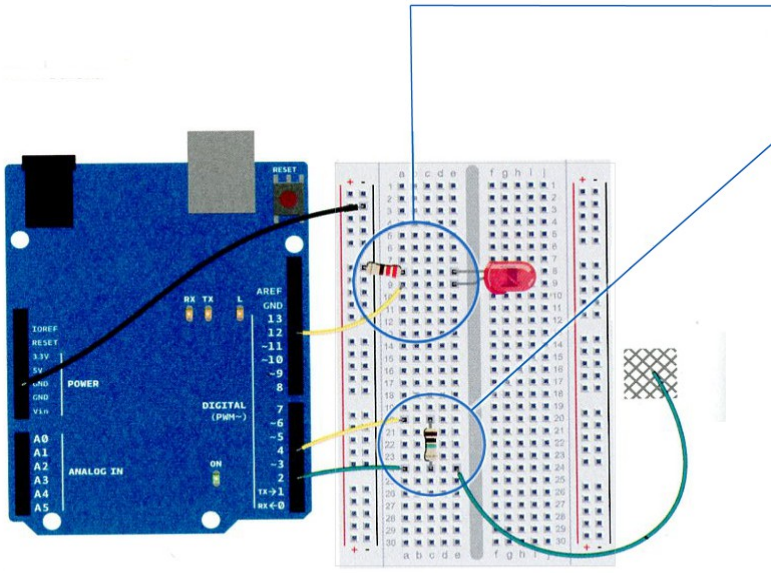


Figura 1

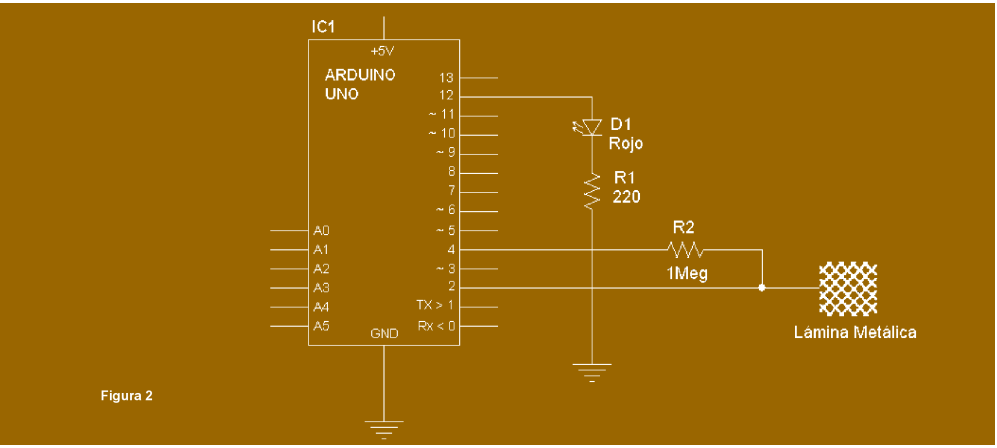


Figura 2

1. Conectar el diodo LED al pin 12 y conectar el cátodo a masa a través de una resistencia de 220 ohmios como se muestra en esta ilustración.

2. Conectar el pin digital 2 y el 4 a la placa de pruebas. Conectar estos dos pines entre sí mediante una resistencia de 1Mega ohmio. En la misma línea de pines de la placa de pruebas en donde se conecta el terminal 2 insertar un cable largo (como mínimo entre 8 y 10 centímetros) que salga fuera de la placa de pruebas. El otro extremo conectarlo a una lámina de aluminio como pueda ser un trozo de papel de aluminio. Este será el sensor.

No es necesario suministrar una tensión de +5V a la placa de pruebas en este proyecto. El pin digital 4 suministra la energía al sensor.

Al igual que con otros proyectos con LEDs, se puede poner un difusor encima del LED para conseguir una luz mucho más atractiva. Una bola de pin-pon o una pantalla para lámpara fabricada con papel o plástico pueden valer perfectamente como difusores y funcionarán perfectamente.

Se puede ocultar el sensor detrás de algo solido y ver si seguirá funcionando. La capacidad se puede medir a través de un material no conductor como la madera, el plástico o el cristal. Si se aumenta el tamaño del sensor de forma que su superficie metálica aumente también se incrementará la sensibilidad del circuito: se puede conectar una hoja de aluminio de mayor tamaño, o una placa de cobre al final del cable. Se podría montar el diodo LED al final de un cilindro de plástico con el difusor encima y a continuación fabricar con una madera fina un base recubierta por su parte interior con papel de aluminio conectando el cable de la placa de pruebas a este papel. De esta manera se fabricará una lámpara cuya base puede funcionar como un sensor al tacto. Actualizar la variable del umbral dentro del código cuando se modifique el tamaño del sensor para asegurarse que el circuito funciona correctamente.

EL CÓDIGO

Importar la librería CapacitiveSensor

Al comienzo del programa incluir la librería CapacitiveSensor. Se hace de la misma forma que se hizo al incluir la librería **Servo** en el proyecto "Medidor del estado de ánimo" en la página 62.
Crear una petición con el nombre de la librería. Cuando se usa esta librería y mediante esta petición se le indica que pins serán usados para enviar y recibir información. En este caso, el pin 4 envía información al sensor formado por una lámina metálica y a través de la resistencia de 1Mega ohmio, y el pin 2 es el pin que recibe la información.

Configurar el umbral

Configurar una variable para el umbral de detección el cual encenderá la lámpara un vez superado este umbral. Deberá de cambiar este valor del umbral después de comprobar el valor que permite encender el LED al tocar el sensor.
En esta línea se define el pin del LED para poder encenderlo y apagarlo.

En el bloque de la función **setup()**, abrir el monitor serie a una velocidad de 9600 baudios por segundo. Esto se hace para poder ver los valores de lectura del sensor en la ventana del monitor serie. Aquí también se define el pin del led como **OUTPUT** (salida).

Configurar la sensibilidad al tacto

Dentro de la función **loop()**, crear una variable del tipo long para guardar el valor del sensor. La librería devuelve el valor del sensor usando un comando llamado **capacitiveSensor()** que toma un argumento que identifica el número de muestras que se van a leer (30). Si se leen unas pocas de muestras, será posible ver una gran variación de valores en el sensor. Si se indican demasiadas muestras, será necesario introducir un retardo para que le de tiempo a leer tantas veces. 30 muestras es un buen valor para comenzar. A continuación se muestra el valor que se lee del sensor en la ventana del monitor serie.

Control de la lámpara

Con una instrucción **if()....else,** se comprueba si el valor leído del sensor es mayor que el valor del umbral. Si es así, se enciende el diodo LED rojo. Si no es así, se apaga el LED.

Al final se añade un pequeño retardo con la instrucción **delay(100)** antes de finalizar la función **loop()**.

```
#include <CapacitiveSensor.h>
CapacitiveSensor capSensor = CapacitiveSensor(4,2);
```

```
int Umbral = 1000;
const int PinLed = 12;
```

```
void setup() {
 Serial.begin(9600);
  pinMode(PinLed, OUTPUT);
 }
```

```
void loop() {
 long ValorSensor = capSensor.capacitiveSensor(30);
 Serial.print(ValorSensor);
```

```
 if(ValorSensor > Umbral){
   digitalWrite(PinLed, HIGH);
 }
 else{
   digitalWrite(PinLed, LOW);
 }
```

```
 delay(100);
}
```

COMO SE UTILIZA

Después de programar la tarjeta Arduino será necesario averiguar que valores son los que encienden el diodo LED al tocar la lámina metálica (sensor). Abrir el monitor serie y anotar los valores que aparecen cuando no se toca el sensor. Tocar suavemente sobre la lámina metálica que hace de sensor, debe de ver como aumentan los valore mostrados en el monitor serie. Tocar haciendo mayor presión y los valores deberán de aumentar mucho más.

Una vez que tenga una idea del rango de valores que produce el sensor según el nivel de presión ejercido, volver al programa (sketch) y cambiar el valor de la variable del umbral a un número ligeramente por encima cuando no se toca el sensor, pero con un valor menor que el que aparece cuando se toca. Por ejemplo, si el valor mostrado sin tocar varia entre 0 y un valor máximo de 20, y cuando se toca sin presionar varia entre 1300 y más de 2000, se puede dejar el valor del umbral como está.

Es posible hacer que la lámpara se encienda sin tocar la lámina metálica, ya que sin tocar (valores entre 0 y 15) los valores son muy bajos y si se acerca un dedo a menos de unos milímetros de la placa los valores aumentan a un mínimo de 23 (puede variar). Por tanto si ahora se pone el valor del umbral a 30 será posible encender la lámpara sin tocar el sensor. Cargar el sketch con este nuevo valor del umbral. El diodo LED deberá de encenderse si se aproxima un dedo a pocos milímetros del sensor sin tocarlo y deberá de apagarse cuando se aleje. Si no consigue que se encienda el LED puede disminuir el valor del umbral un poco (de 30 a 20).

Habrá observado que los valores mostrados en el monitor serie varían según el nivel de presión ejercido sobre el sensor. ¿Puede usar estos valores para conseguir otras interacciones con el diodo LED ? ¿Se podrán usar múltiples sensores para variar gradualmente la luz del LED hasta que se consiga el mayor brillo posible y para apagar la luz la suavemente? Si coloca diferentes valores de resistencia entre los pins 2 y 4 conseguirá cambios en la sensibilidad. ¿Es esto útil para su interface?

Las librerías de terceros como la CapacitiveSensor de Paul Badger son herramientas útiles para aumentar las capacidades de Arduino. Una vez instaladas, se comportan de la misma forma que las que se incluyen con el software de Arduino.

14

POTENCIÓMETRO

INGREDIENTES

RETOCAR EL LOGOTIPO DE ARDUINO

USANDO LA COMUNICACIÓN SERIE, SE VA A USAR ARDUINO PARA CONTROLAR UN PROGRAMA EN UN ORDENADOR

Descubra: comunicación serie con un programa de ordenador, Processing

Tiempo: **45MINUTOS**

Nivel: **alto**

Proyectos en los que se basa: **1,2,3**

Se han realizado un montón de cosas interesantes con el mundo físico, ahora es el momento de controlar un ordenador usando Arduino. Cuando se programa Arduino, se está abriendo una conexión entre el ordenador y el microcontrolador de la placa Arduino. Se puede usar esta conexión para enviar datos de ida y vuelta a otras aplicaciones.

Arduino tiene un chip que convierte la comunicación USB del ordenador en una comunicación serie que Arduino puede usar. La comunicación serie permite que dos equipos electrónicos, una tarjeta Arduino y un PC, puedan intercambiar bits de información en serie, es decir, uno tras otro en el tiempo.

Para poder establecer una comunicación serie entre dos dispositivos es necesario que la velocidad de transmisión de uno a otro, y al revés, sea la misma. Probablemente habrá observado cuando ha usado el monitor serie que en la parte inferior derecha de la ventana aparece un número. Ese número, 9600 bits por segundo, o baudios, es el mismo valor que se ha declarado usando la instrucción **Serial.begin()**. Esa es la velocidad a la cual Arduino y el ordenador intercambian datos. Un bit es la unidad mínima de información que un ordenador puede entender.

Se ha usado el monitor serie para examinar los valores de las entradas digitales; se usará un método similar para obtener valores con un programa que es un entorno de programación diferente al IDE de Arduino y que se conoce con el nombre de **Processing**. Processing está basado en Java, el entorno de programación de Arduino está basado en Processing. Ambos son muy similares, así que no debería tener ningún problema para usarlo.

Antes de comenzar con el proyecto, descargar la última versión de Processing de la página ***processing.org***. Le puede resultar útil mirar el apartado de “Getting started” (Comenzando) y echar un vistazo los tutoriales de la página ***processing.org/learning***. Toda esta información le ayudará a familiarizarse con Processing antes de comenzar a escribir el software para comunicarse con Arduino.

La forma más eficiente de enviar datos entre Arduino y Processing es usando la función **Serial.write()** en Arduino. Es similar a la función **Serial.print()** que se va a usar para enviar información desde Arduino al ordenador que este conectado, pero en lugar de enviar información que se puede leer por los humanos como son números y letras, se enviarán valores entre 0 y 255, como un grupo de bytes. Esto limita los valores que Arduino puede enviar, pero permite una transmisión más rápida de la información.

Tanto en el ordenador como en Arduino, existe algo llamado buffer serie el cual almacena en su interior información hasta que el programa la necesite. Se enviarán bytes desde Arduino al buffer serie de Processing. Cuando el programa lee la información que contiene el buffer, se vacía el espacio del buffer para guardar más información que le pueda enviar el programa Arduino.

Cuando se está usando una comunicación serie entre dispositivos y programas, es muy importante que ambos sepan cual será la velocidad de comunicación entre ellos, y además el tipo de información que deberían estar esperando. Cuando se encuentra con alguien que conoce, probablemente esperará que le diga “Hola”, si por el contrario le dice algo como “El gato está con el pelo erizado”, su respuesta le cojera por sorpresa. Con el software ocurre lo mismo, será necesario que ambos programas estén de acuerdo sobre lo que se enviá y sobre que se recibe.

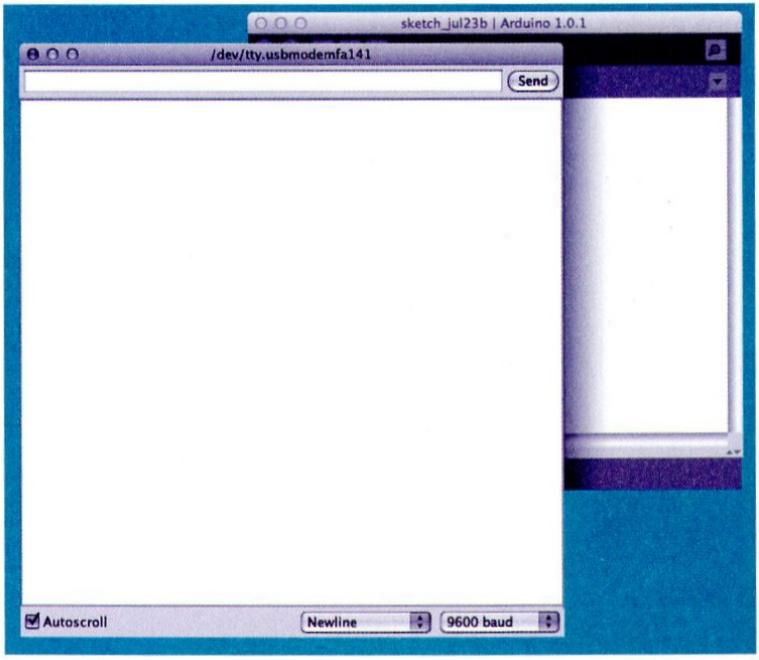


Figura 1

MONTANDO EL CIRCUITO

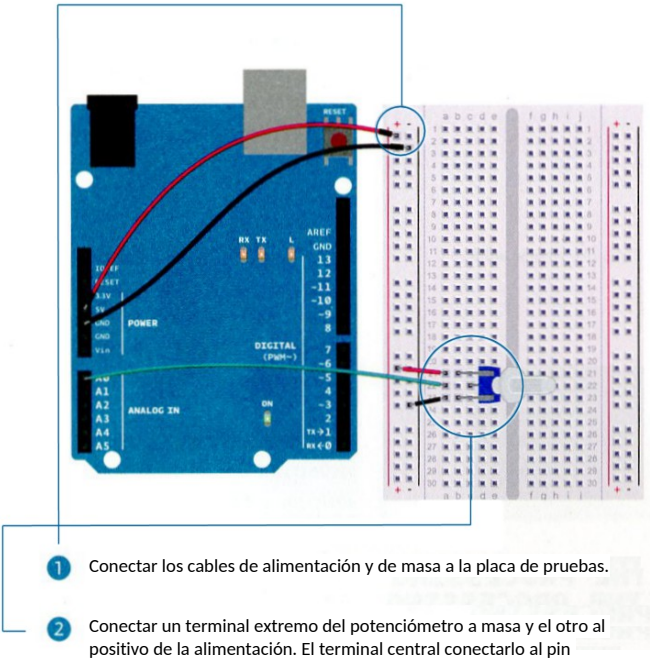


Figura 2

1. Conectar los cables de alimentación y de masa a la placa de pruebas.

2. Conectar un terminal extremo del potenciómetro a masa y el otro al positivo de la alimentación. El terminal central conectarlo al pin analógico 0 de la placa Arduino.

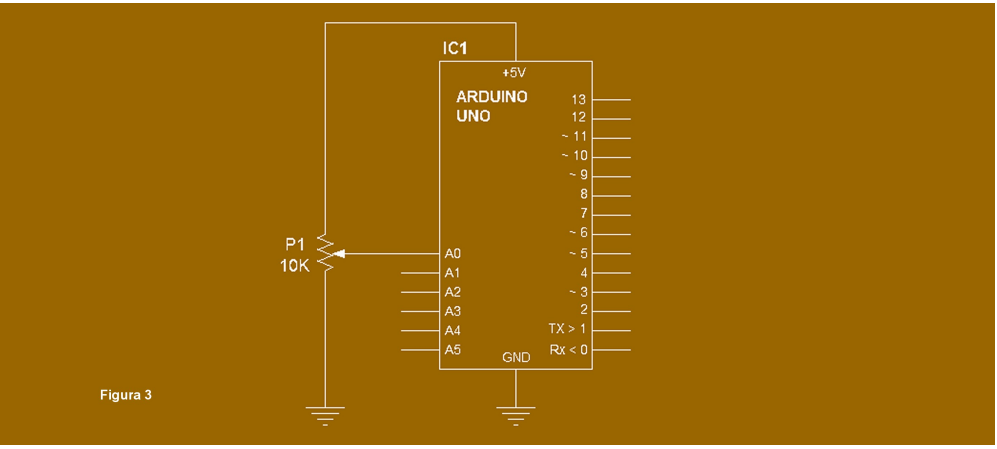


Figura 3

EL CÓDIGO DE ARDUINO

Abrir una conexión serie

Primero, programar su tarjeta Arduino. Dentro del **setup()**, se configura la comunicación serie, tal como se hizo en otros proyectos cuando se quería mostrar los valores del sensor conectado a la placa. En el programa Processing se escribirá la misma velocidad de transmisión serie que se escribió en Arduino.

Enviar el valor del sensor

Dentro de **loop()**, se va a usar la instrucción **Serial.write()** para enviar información a través de la conexión serie entre Arduino y el ordenador. **Serial.write()** solo puede enviar valores entre 0 y 255. Para asegurarse que se van a enviar valores dentro de este rango, dividir el valor de la lectura analógica por 4.

Permitir que el ADC se estabilice

Después de enviar un byte, esperar durante un millisegundo para darle tiempo al convertidor analógico-digital (ADC) a realizar esta conversión. Cargar el programa en la tarjeta Arduino y a continuación dejar el IDE de Arduino mientras se escribe el sketch dentro del programa Processing

EL CÓDIGO DE PROCESSING

Importar la configuración del objeto serie

El lenguaje de Processing es similar al lenguaje de Arduino, pero hay bastantes diferencias que deberá de buscar dentro de los tutoriales de este programa y dentro de la guía “Getting Started” mencionada con anterioridad para conseguir familiarizarse con este lenguaje.
Abrir una nuevo sketch de Processing. Processing, a diferencia de Arduino, no sabe trabajar con los puertos sin que se incluya una librería externa. Importar la librería serie. Es necesario crear un copia del objeto serie, tal como se hizo en Arduino con la librería Servo. Se usará este nombre único siempre que se necesite usar la conexión serie.

Crear un objeto para la imagen

Para usar imágenes en Processing, es necesario crear un objeto que contendrá la imagen y darle un nombre.

```
1  void setup() {
2   Serial.begin(9600);
3  }
```

```
4  void loop() {
5   Serial.write(analogRead(A0)/4);
```

```
6   delay(1);
7  }
```

GRABAR Y CERRAR
EL IDE DE ARDUINO
AHORA.
VAMOS A EMPEZAR

```
1  import processing.serial.*;
2  Serial miPuerto;
```

```
3  PImage logotipo;
```

Variable para guardar el color de fondo

Crear una variable que guardará el matiz del color de fondo del logo de Arduino. El logo es un fichero de imagen png, y tiene transparencia, así que es posible ver como cambia el color de fondo.

Processing tiene una función **setup()**, al igual que Arduino. Aquí es donde se abrirá la conexión serie además le darle al programa un par de parámetros los cuales serán usados mientras se ejecuta el sketch.

Configurando el modo de color

Se puede cambiar la forma en que Processing trabaja con la información de color. Típicamente, trabaja con tres canales de color, Rojo, Verde y Azul (RGB). Es similar a la forma en como se mezclaron los colores en el proyecto 4 (lámpara de mezcla de colores, página 52), cuando se usaron valores entre 0 y 255 para cambiar el color del LED RGB. En este programa, se va a usar un modo de color llamado HSB, el cual trabaja con el Matiz, con la Saturación y el Brillo de una imagen. Se va a cambiar el matiz del color de fondo al girar el mando de un potenciómetro.

La instrucción **colorMode()** necesita dos argumentos: el tipo de modo, y el valor máximo que puede alcanzar.

Cargando la imagen

Para cargar el logo de Arduino dentro del sketch, leerlo en el objeto logotipo que se ha creado con anterioridad, Cuando se proporciona la URL de una imagen, Processing la descargará cuando se ejecute el programa.
Con la función **size()**, se le indica a Processing el tamaño de la ventana de visualización que contiene la imagen. Si se usan **logo.width** y **logo.height** como argumentos, el sketch escala automáticamente el tamaño de la imagen que se está usando.

Mostrando los puertos serie disponibles

Processing tiene la capacidad de mostrar mensajes de status usando la instrucción **println()**. Si se usan en conjunción con la función **Serial.list()**, se conseguirá un listado de todos los puertos serie que tiene el ordenador cuando el programa se inicie por primera vez. Se usará esta información una vez se haya finalizado la programación para saber a que puerto está conectada la tarjeta Arduino.

Creando un objeto serie

Es necesario indicarle a Processing en que puerto está conectada la tarjeta Arduino. Para que el objeto llamado **miPuerto** pueda almacenar la información que necesita, el programa también necesita saber que es una copia del objeto serie. Los parámetros que hay que indicar son con que aplicación se va a hablar, a través de qué puerto serie se va a comunicar y a que velocidad lo hará.

```
int colordefondo = 0;

void setup() {

  colorMode(HSB, 255);

  logotipo = loadImage("http://arduino.cc/logo.png");
  size(logotipo.width, logotipo.height);

  println("Puertos serie disponibles:");
  println(Serial.list());

  miPuerto = new Serial(this, Serial.list()[0], 9600);
}
```

El atributo **this** le dice a Processing que se va a usar una conexión serie con esta aplicación. El argumento de la instrucción **Serial.list() [0]** especifica cual será el puerto que se va a usar. **Serial.list()** contiene una matriz de todos los dispositivos serie que están conectados. El argumento **9600** debe de resultarle familiar, define la velocidad a la cual el programa se va a comunicar.

La función **draw()** es análoga a la función **loop()** de Arduino en donde el programa se ejecuta dentro de un bucle continuo. Aquí es donde se realizan los dibujos en la ventana del programa

Leyendo los datos de Arduino mediante el puerto serie

Comprobar si Arduino ha enviado información. El comando **miPuerto.available()** le dice al programa si hay algo en el buffer serie. Si hay bytes allí, lee estos valores y los guarda dentro de la variable **colordefondo** además de mostrarla en la ventana del programa.

Configurando el fondo de la imagen y mostrando la imagen

La función **background()** establece el color de fondo de la ventana. Dispone de tres argumentos. El primer argumento es el matiz, el siguiente el brillo, y el último la saturación. Usar la variable colordefondo como valor del matiz, y colocar el brillo y la saturación a su valor máximo, 255.
Se dibujará el logotipo de Arduino con la instrucción **image()**. Es necesario decirle a esta instrucción **image()** que se va a dibujar, y a partir de qué coordenadas de la ventana se comienza a realizar el dibujo. Las coordenadas 0,0 se localizan en la parte superior izquierda de la ventana del programa, para comenzar a dibujar allí.

CÓMO SE UTILIZA

Conectar la placa Arduino y abrir el monitor serie. Girar el mando del potenciómetro de la placa de pruebas. Debe de verse un número de caracteres cuando se gira el mando. En el monitor serie se esperan caracteres ASCII, no bytes en bruto. ASCII es un tipo de información codificada para representar texto en los ordenadores. Lo que ve en la ventana es el monitor serie tratando de interpretar los bytes como código ASCII.

Cuando se utiliza **Serial.println()**, se enviá información con formato para el monitor serie. Cuando se utiliza **Serial.write()**, como en esta aplicación que se esta realizando ahora, se está enviando información en bruto. Programas como Procesing puede entender estos bytes en bruto.

```
13 void draw() {
```

```
14  if (miPuerto.available() >0) {
15   colordefondo = miPuerto.read();
16   println(colordefondo);
17  }
```

```
18  background(colordefondo, 255, 255);
19  image(logotipo,0 ,0);
20 }
```

Cerrar el monitor serie y mantener abierto el IDE de Arduino. Abrir el programa Processing y cargar el sketch, ejecutarlo al presionar el botón con forma de flecha dentro del IDE de Processing. Mirar la ventana de salida de Processing (parte inferior). Debe de verse una imagen similar a la mostrada en la figura inferior (si trabaja con Windows).

Esta ventana de salida muestra un listado de todos los puertos serie disponibles en el ordenador. Si usa un sistema operativo como pueda ser OSX, buscar una cadena de texto en donde aparezca algo como "/dev/tty.usbmodem411", los más probable es que sea el primer puerto disponible de la lista. En Linux, puede aparecer como "/dev/ttyUSB0" o similar. En Windows (como en la imagen superior), aparecerán como puertos COM, el mismo que se uso con el IDE de Arduino y alguno más; en este caso los puertos COM1 y COM11. El número que aparece delante de **Serial.list ()[0]** es el número de puerto que se va a usar (0 en este caso). Cambiar este número en el sketch de Processing para que coincida con el puerto de comunicaciones del ordenador. En este caso será el puerto COM11. Por tanto la instrucción del sketch de Processing en Windows quedará de esta forma " **miPuerto = new Serial(this, Serial.list()[11], 9600);**".

Resetear el sketch de Processing y volver a ejecutarlo. Cuando el programa comience a funcionar se abrirá otra ventana con el logotipo de Arduino con un fondo rojo, girar el mando del potenciómetro sobre la placa de pruebas conectada a Arduino. Al hacerlo el color de fondo de la ventana del logotipo cambiará de color. A la vez en la ventana de salida de Processing se podrán ver como cambian los valores al girar el mando de este potenciómetro. Estos números se corresponden con los bytes en bruto que se envían desde Arduino.

Una vez que haya girado el potenciómetro y obtenido un color de fondo según el deseo de su corazón, intente reemplazar el potenciómetro por un sensor analógico (detector de luz, temperatura, etc). Encontrar algo que le parezca interesante para controlar el color de fondo. ¿Qué le parece esta interacción? Es probablemente diferente cuando se usa un ratón o un teclado ¿cual es le gusta más de los dos?

Cuando se usa la comunicación serie, solo una aplicación se puede comunicar con Arduino en ese momento. Así que si se está ejecutando el sketch de Processing que está conectado a Arduino, no será posible cargar un nuevo sketch en Arduino o usar el monitor serie del IDE de Arduino hasta que se pare el sketch de Processing.

Con Processing y otros entornos de programación, puede controlar la comunicación con el ordenador de una forma llamativa y novedosa. Si se muestra interesado acerca de las posibilidades de controlar los contenidos de su ordenador, puede dedicarle algún tiempo a la experimentación con Processing. Existen varios ejemplos de comunicación serie tanto en el IDE de Arduino como en el IDE de Processing que le ayudarán a saber más sobre este tema.

La comunicación serie permite a Arduino comunicarse con los programas de un ordenador. Processing es un entorno de programación de código abierto en el que se basa el IDE de Arduino. Es posible controlar un sketch de Processing mediante la comunicación serie con Arduino.

15

OPTOACOPLADOR

RESISTENCIA DE 220 OHMIOS

INGREDIENTES

HACKEAR BOTONES

OBTENER EL CONTROL DE OTRO COMPONENTE QUE TENGA CERCA. USANDO ALGUNA CIRCUITERÍA ADICIONAL, PUEDE “PRESIONAR” BOTONES CON ARDUINO

Descubra: optoacoplador, conexión con otros componentes

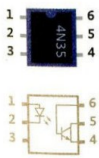
Tiempo: **45MINUTOS**

Nivel: **alto**

Proyectos en los que se basa: **1,2,9**

Advertencia: Ya no será mas un principiante si esta realizando este proyecto. Abrirá un dispositivo electrónico (mando a distancia, teclado, etc) y lo modificará. Perderá la garantía al abrirlo, y si no tiene cuidado podría estropearlo. Debe de asegurarse de que todos los conceptos básicos de la electricidad y de la electrónica que se han visto en los proyectos anteriores le resultan familiares antes de realizar este proyecto. Le recomendamos que utilice componentes y dispositivos electrónicos baratos en los proyectos que realice, y que no le importe estropear, hasta que adquiera la experiencia y confianza necesaria en el montaje de proyectos electrónicos.

En tanto que Arduino puede controlar un montón de cosas, algunas veces es más fácil usar herramientas que han sido creadas para propósitos específicos. Quizás quiera controlar una televisión o un reproductor de música, o conducir un coche de control remoto. Muchos de los dispositivos electrónicos disponen de un mando de control con botones, y muchos de esos botones se pueden hackear de manera que pueda “presionarlos” usando Arduino. El control de una grabadora de sonido es un buen ejemplo. Si quiere grabar y después reproducir el sonido grabado, le podría costar un gran esfuerzo conseguir que Arduino haga esto. Es mucho más fácil obtener un pequeño dispositivo que grabe y reproduzca sonidos, y reemplazar sus botones por las salidas controladas por Arduino.



Los **optoacopladores** son circuitos integrados que permiten controlar un circuito desde otro circuito diferente y sin ninguna conexión eléctrica directa entre los dos. Interiormente un optoacoplador de este tipo dispone de un diodo LED (circuito de control) y un foto transistor que actuá como un detector de luz (circuito a controlar). Cuando el diodo LED interno del optoacoplador se enciende mediante Arduino, el detector de luz cierra un interruptor interno (transistor). El interruptor interno esta conectado a dos terminales de salida del optoacoplador (terminales 4 y 5). Cuando este interruptor esta cerrado, los dos terminales de salida se conectan entre sí. Cuando el interruptor esta abierto, los terminales 4 y 5 están desconectados. De esta manera, es posible cerrar interruptores de otros dispositivos sin una conexión eléctrica directa con Arduino, usando siempre un optoacoplador.

En este proyecto, los diagramas son para controlar un módulo de grabación digital que permite grabar y reproducir un sonido durante 20 segundos, pero esta idea es también válida para cualquier dispositivo electrónico que disponga de un interruptor al cual se pueda acceder. Mientras sea posible para realizar este proyecto se recomienda hacerlo sin soldar ningún cable, así será mucho más fácil su realización, en caso contrario puede abrir el siguiente enlace para saber cómo soldar: ***arduino.cc/soldering***

MONTANDO EL CIRCUITO

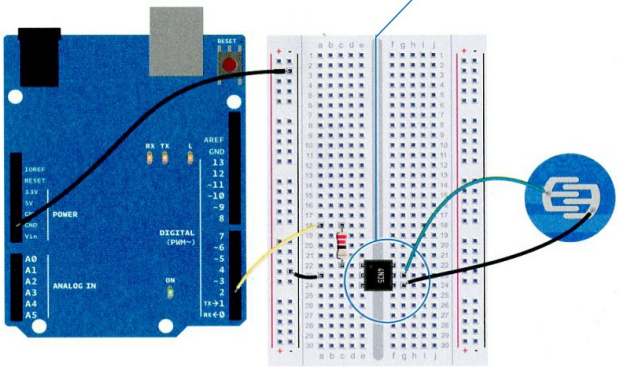


Figura 1

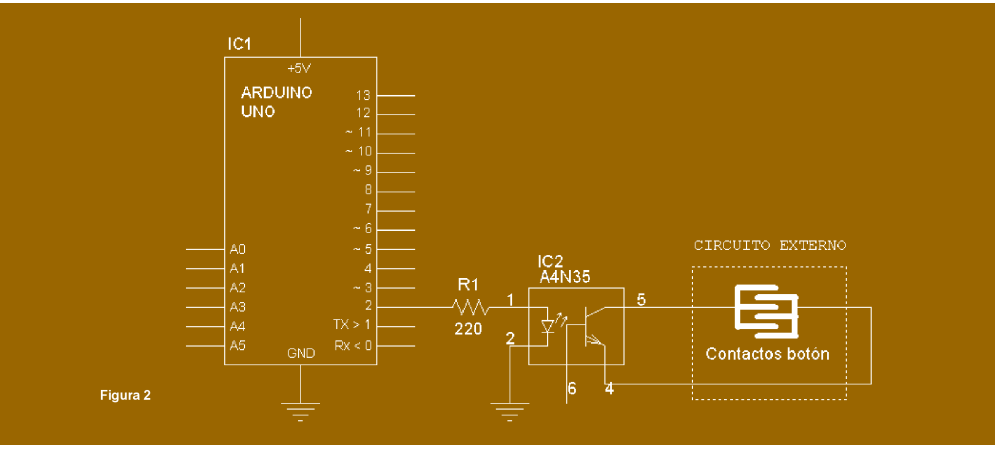


Figura 2

1. Conectar la masa a la placa de pruebas a través de Arduino.

2. Colocar el optoacoplador sobre la placa de pruebas de manera que quede colocado en el centro de la placa (ver esta ilustración)

3. Colocar el pin 1 del optoacoplador al terminal 2 de Arduino en serie con una resistencia de 220 ohmios (recordad, se alimenta un diodo LED en el interior así que hay que conectar le una resistencia para que no se queme). Conectar el pin 2 del optoacoplador a masa.

4. Sobre la placa del módulo de sonido hay un determinado número de componentes electrónicos, incluyendo el botón de reproducción.

   Para controlar el interruptor del módulo de sonido, es necesario quitar el botón. Darle la vuelva al circuito y encontrar las lengüetas que sujetan al botón en su lugar. Doble con cuidado las lengüetas y quite el botón de la placa.

5. Debajo del botón hay dos pequeñas láminas metálica. Este diseño es típico de muchos dispositivos electrónicos que tienen botones que se presionan. Las dos “horquillas” de este diseño son los dos lados del interruptor. Un pequeño disco de metal en el interior del botón de presión conecta estas dos horquillas metálicas cuando se presiona el botón.

6. Cuando las horquillas metálicas se conectan entre sí, el interruptor esta cerrando el circuito de la placa. Se cerrará este interruptor usando el optoacoplador.

   Esto método, cerrar un interruptor usando un optoacoplador, trabaja solo si uno de los dos lados del interruptor del pulsador de presión esta conectado a masa del módulo de sonido. Si no está seguro de que su módulo este conectado de esta forma, usar un polímetro y medir la tensión entre una de las pestañas y masa de su dispositivo. Es necesario hacer esto con el circuito alimentado, así hay que tener cuidado de no hacer un cortocircuito con las puntas del polímetro al tocar en algún lugar de la placa.

   Una vez que sepa qué horquilla está conectada a masa, podrá quitarle alimentación al circuito.

7. Lo siguiente que debe de hacer es conectar un cable a cada una de las pequeñas láminas metálicas u horquillas. Si va a soldar unos cables, tenga cuidado de no unir accidentalmente estas dos pestañas con un punto de estaño, de manera que el interruptor quede permanentemente cerrado. Si no a va a soldar y va a usar pegamento, asegurarse que las conexiones está bien realizadas, ya que si no el interruptor no se cerrará. Al final puede probar con un polímetro en la escala de resistencia que el valor en ohmios entre los cables unidos a las horquillas es prácticamente infinito. En caso de que se obtenga un valor inferior a 1 ohmio significa que existe un cortocircuito entre las láminas metálicas.

8. Conectar los dos cables unidos a las láminas metálicas de su módulo de sonido a los terminales 4 y 5 del optoacoplador. Conectar el cable del interruptor que esta conectado a masa al terminal 4 del optoacoplador. Conectar el otro cable al terminal 5 del optoacoplador.

EL CÓDIGO

Crear una constante

La parte más interesante de este proyecto se encuentra en el módulo de sonido y en el optoacoplador. Este código es similar al primer proyecto con código que hizo con Arduino. Se va a reproducir un sonido cada 20 segundos al poner el pin 2 de Arduino en estado **HIGH** (alto).
Crear una constante para el pin de control de optoacoplador.

Configurar la dirección del pin

En el **setup()**, establecer el pin del optoacoplador como una salida.

Poner el pin en estado alto y bajo

En el bloque de la función **loop()**, se pone en estado **HIGH** el pin (PinOpto) de Arduino durante unos pocos mili segundos, el tiempo suficiente como para cerrar el interruptor del módulo de sonido. A continuación se pone en estado **LOW** (bajo).

Esperar un poco de tiempo

Esperar durante 21 segundos para que todo el mensaje grabado se pueda reproducir antes de comenzar de nuevo con **loop()**.

COMO SE UTILIZA

Conectar la alimentación del módulo de sonido. Presionar y mantener el botón de grabación de este módulo. Mientras el botón está presionado se puede realizar una grabación de audio usando el micrófono que incorpora. Usar su voz, de un gato, o usar las ollas y las sartenes de la cocina para hacer algo de ruido (ser cuidadoso con el gato).

Una vez que este satisfecho con la grabación realizada, darle alimentación a la placa Arduino usando el cable USB y conectándolo a un ordenador. Su grabación deberá comenzar a sonar. Si la grabación dura 20 segundos comenzará de nuevo a funcionar unos segundos después de que finalice.

Intente experimentar con diferentes sonidos y tiempos de grabación cambiando el tiempo de reproducción mediante la instrucción delay() del programa.
Si se cierra el interruptor de reproducción mientras un sonido se está reproduciendo, se parará.
¿Cómo se puede usar esto para crear secuencias únicas de sonidos?

```
1  const int PinOpto = 2;
```

```
2  void setup() {
3   pinMode(PinOpto, OUTPUT);
4  }
```

```
5  void loop() {
6   digitalWrite(PinOpto, HIGH);
7   delay(15);
8   digitalWrite(PinOpto, LOW);
```

```
9   delay(21000);
10 }
```

Los circuitos integrados están prácticamente en todos los equipos electrónicos. El chip grande de 28 pins de la placa Arduino es un circuito integrado o IC. Existen otros ICs en la placa Arduino que le dan soporte a este chip de 28 pins, permitiendo que se comunique con el PC y que pueda disponer de una tensión de alimentación estabilizada. El optoacoplador y el chip principal de Arduino son integrados con encapsulados llamados **Dual In-Line Package (DIP)** (encapsulado en doble línea). Estos chips tipo DIP son los que la mayoría de los aficionados a la electrónica utilizan por que son más fáciles de colocar en la placa de pruebas y además no hay que soldarlos para poder usarlos.

El ejemplo del proyecto solo reproduce el mismo sonido o sonidos cada intervalo regular de tiempo. ¿Cómo se podría incorporar unas entradas como en proyectos anteriores para activar estos sonidos en un determinado momento? ¿qué otras cosas que se alimentan mediante una batería y que tenga por casa se podrían controla mediante Arduino? Esta técnica de controlar un dispositivo electrónico usando un optoacoplador y conectando los dos terminales de un interruptor puede ser usado en muchos otros dispositivos. ¿Qué otros dispositivos quiere usted controlar?

Los optoacopladores pueden controlar dispositivos que se localizan en un circuito diferente (otro equipo electrónico). Los dos circuitos están eléctricamente separados el uno del otro y en el interior del optoacoplador

A/Z

Aceleró-metro
Actuador
Aislante
Amperaje (amperios)
Analógico
Ánodo
Argumento
Baudio
Binario
Bit
Booleano
Buffer serie
Byte
Calibración
Capacidad
Carga
Cátodo
Circuito
Circuito integrado (IC)
Comunicación serie
Condensadores de desacoplo
Conductor
Constante
Convertidor Analógico-a-Digital (ADC)
Copia
Corriente
Corriente alterna
Corriente continua
Cortocircuito
Depuración
Digital
Divisor de voltaje o de tensión
Drenador (transistor)
Electricidad
Encapsulado en doble línea (DIP)
Flotante
Foto célula
Foto resistencia
Foto transistor
Fuente (transistor)
Fuente de alimentación
Función
Hoja de datos
IDE
Index
Inducción
Int
Interruptor
LED de cátodo común
Ley de Ohm
Librería
Long
Masa
Matríz
Microcontrolador
Mili segundo
Modificador
Modulación por Ancho de Pulso (PWM)
Monitor serie
Objeto
Ohm
Onda cuadrada
Optoacoplador
Paralelo
Parámetro
Periodo
Polarizado
Processing
Pseudocódigo
Puerta (transistor)
Relación cíclica
Resistencia
Sensor
Serie
Sin signo
Sketch
Soldar
Tensión inversa
Transductor
Transistor
USB
Variable
Variable global
Variable local
Voltaje

GLOSARIO

HAY UNA SERIE DE TÉRMINOS NUEVOS QUE HA APRENDIDO AL REALIZAR ESTOS PROYECTOS. LOS HE RECOGIDO TODOS AQUÍ PARA CREAR UN GLOSARIO

A

Aceleró-metro: Se trata de un sensor que mide la aceleración. Algunas veces también se usan para detectar la orientación o la inclinación.

Actuador: Un tipo de componente electrónico que convierte la energía eléctrica en movimiento. Los motores son un tipo de actuador.

Aislante: Un tipo de material que evita que la corriente eléctrica pueda circular. Los materiales conductores como los cables se cubren con materiales aislantes como el plástico.

Amperaje (amperios): La cantidad de carga eléctrica que circula por un circuito de un punto a otro del mismo. Indica la cantidad de intensidad (corriente por unidad de tiempo) que fluye a través de un conductor, como pueda ser un cable.

Analógico: Algo que puede variar continuamente en el tiempo entre sus valores máximo y mínimo tomando cualquier valor entre estos extremos. Por ejemplo, una tensión que varía de 0 a 5V se puede tomar cualquier valor intermedio (1,2V, 3.123V, etc).

Ánodo: El terminal positivo de un diodo (recordar que un LED es un tipo de diodo).

Argumento: Un tipo de dato que se le suministra a una función como una entrada. Por ejemplo, al usar digitalRead() hay que indicar que pin se va a chequear, en este caso se le suministra el argumento con el formato de un número digitalRead(7), o una variable númerica, digitalRead(PinPulsador).

B

Baudio: Término que se refiere a “bits por segundo”, indicando la velocidad a la cual un ordenador se comunica con otro.

Binario: Un sistema que solo dispone de dos estados, como verdadero/falso o sin tensión/con tensión o “0”/”1”.

Bit: La pieza más pequeña de información que un ordenador puede enviar o recibir. Tiene dos estados, 0 y 1. Cero quiere decir que no tiene tensión y 1 que tiene tensión.

Booleano (boolean): Variables que indican si algo es verdadero o falso.

Buffer serie: Un lugar de la memoria de un ordenador o de un microcontrolador donde la información recibida en una comunicación serie es guardada hasta que es leída por un programa.

Byte: Son 8 bits de información. Un byte puede almacenar cualquier número entre 0 y 255. También se conoce con el nombre de palabra digital.

C

Calibración: El proceso por el cual se realizan ajustes en algunos números o en algunos componentes (potenciómetros, resistencia ajustables, etc) para conseguir que un programa o un circuito electrónico funcione mejor. En los proyectos de Arduino esto se hace a menudo cuando se usan sensores para medir parámetros físicos en diferentes circunstancias, por ejemplo, al usar una foto resistencia que mide el nivel de luminosidad de una habitación.

Capacidad: La habilidad de algo para almacenar energía eléctrica (carga eléctrica). Esta carga se puede medir con la librería del Sensor Capacitivo, como en el proyecto número 13 (página 136). Los componentes electrónicos que usan principalmente esta propiedad son los condensadores.

Carga: Cualquier dispositivo o componente electrónico que transforme la energía eléctrica en calor, luz, movimiento, sonido,etc. Por ejemplo, una resistencia transforma la energía eléctrica en calor.

Cátodo: El terminal de un diodo que normalmente se conecta a masa.

Circuito: El conjunto de componentes electrónicos a través de los cuales la intensidad de la corriente eléctrica circula, saliendo del positivo de la alimentación y volviendo al negativo de esta alimentación y a través de los componentes que forman el circuito. Para que la intensidad circule el circuito deberá de estar cerrado, si existe un interruptor en serie con la alimentación y está abierto el circuito no estará cerrado y la intensidad no circulará.

Circuito integrado (IC): Se trata de un componente electrónico que ha sido creado sobre una pequeña oblea de silicio e integrada dentro de un encapsulado plástico. Los pins, patillas o terminales que sobresalen del encapsulado permiten la interacción con el circuito interno. Muchas veces es muy sencillo usar un IC simplemente sabiendo la función de cada uno de sus terminales sin necesidad de saber como funciona el integrado a nivel interno.

Comunicación serie: El sistema por el cual Arduino se comunica con ordenadores y otros dispositivos. Se trata de enviar un bit de información a la vez en un orden secuencial. La placa Arduino dispone de un convertidor USB a serie el cual permite que se pueda comunicar con otros dispositivos que no tengan un puerto serie dedicado.

Condensadores de desacoplo: Los condensadores que se colocan cerca de los actuadores (relés, transistores, etc) y que son usados para atenuar las variaciones y caídas de tensión que se producen en el momento que los actuadores funcionan. Se colocan cerca de los sensores y actuadores.

Conductor: Algo que permite que la intensidad de la corriente eléctrica pueda circular, como un cable.

Constante (const): Un modificador que convierte a una variable en una constante y que no puede cambiar su valor en un programa, por ejemplo, `const float pi = 3.1415`, la variable (constante con decimales) es pi.

Convertidor Analógico a Digital (ADC): Un circuito que convierte una tensión de una entrada analógica (cualquier valor entre su valor mínimo y máximo) en un número digital que representa esa tensión. Este circuito esta integrado dentro del microcontrolador de Arduino, y está conectado a los pins de entrada analógica A0-A5. Convertir una tensión analógica en un número digital le lleva un poco de tiempo, por eso después de usar la instrucción analogRead() se usa un pequeño retraso para que le tiempo el hacerlo. Con la instrucción delay() se introduce el retraso.

Copia (instance): Una copia de un objeto de software. Se han usado copias de la librería Servo en los proyectos 5 y 12, en cada caso, se ha creado un nombre para la copia de la librería Servo para usar en el proyecto.

Corriente: El movimiento de las cargas eléctricas (electrones) a través de un circuito cerrado. Se mide en culombios. La intensidad (corriente por unidad de tiempo)se mide en amperios.

Corriente alterna: Un tipo de corriente en el cual las cargas eléctricas están cambiando el sentido de circulación periódicamente, primero en un sentido y un tiempo después en el contrario, volviendo de nuevo al sentido inicial. Este es el tipo de electricidad que se encuentran en los enchufes de una vivienda. Hay que matizar que la corriente alterna de un enchufe no se manifiesta hasta que se conecte una carga, como pueda ser una lámpara que este encendida. En el enchufe lo que aparece directamente es una tensión alterna, cuyo valor eficaz son los 220 voltios.

Corriente continua: Un tipo de corriente en el cual las cargas eléctricas siempre circulan en el mismo sentido. Cuando se conecta una bombilla de 9V en paralelo con una pila de 9V se produce este tipo de corriente. En todos los proyectos de Arduino se ha consumido una corriente continua. *Recordad que la corriente se mide en culombios y que la corriente dividida del tiempo o intensidad se mide en amperios*, y se puede medir con un polímetro.

Cortocircuito: Cuando se produce un problema en un circuito eléctrico o electrónico de manera que la intensidad que necesita para funcionar es muy superior a la que fuente de alimentación puede proporcionar y por tanto el circuito deja de funcionar o se quema. Los cortocircuitos se puede producir en varias partes de un circuito, en un componente electrónico como pueda ser un transistor, un condensador, etc. En algunos casos si la fuente de alimentación puede proporcionar una intensidad de cortocircuito el componente que provoca este cortocircuito puede llega a arder.

D

Depuración: El proceso de buscar errores en un circuito o en un código (también referidos como “bugs”), hasta que se consigue que funcione según se espera.

Digital: Un sistema de valores discretos. Como Arduino es un tipo de dispositivo digital, solo trabaja con dos estados discretos, apagado o encendido. Digital significa siempre solo dos estados, binario, cero o uno. Por eso Arduino necesita de un convertidor analógico – digital, para convertir los datos analógicos en binarios (digital).

Divisor de voltaje o tensión: Un tipo de circuito, normalmente formado por dos componentes, que suministra una tensión que es una fracción de la tensión de alimentación a la cual está conectado. En algunos de los proyectos de este libro se han montado divisores de tensión, por ejemplo, cuando se montó una resistencia LDR en serie con una resistencia fija para proporcionar una tensión a una entrada analógica de Arduino (proyecto número 6 Theremin controlado por luz – página 70). Un potenciómetro también es un ejemplo de divisor de tensión.

Drenador: Terminal de un transistor el cual se conecta a una carga (bombilla, altavoz, etc) para ser controlada por este transistor.

E

Electricidad: Un tipo de energía producida por las cargas eléctricas. Se pueden usar componentes electrónicos para transformar la electricidad en otras formas de energía como luz y calor.

Encapsulado en doble línea (DIP): Un tipo de encapsulado usado con los circuitos integrados que permite que estos componentes se puedan insertar con facilidad en una placa de pruebas o en un zócalo montado sobre una PCB.

F

Flotante (float): Un tipo de variable que se puede expresar como una fracción. Esto implica el uso de decimales para los números con coma flotante.

Foto célula: Componente que convierte la energía luminosa en energía eléctrica.

Foto resistencia: Resistencia cuyo valor óhmico varia según la cantidad de luz que recibe, a mayor luz menor resistencia y viceversa. También se conocen con el nombre de Resistencias Dependientes de la Luz o L.D.R.

Foto transistor: Transistor que además de poderse controlar mediante una corriente de base también se puede controlar por luz, ya que dispone de una superficie de cristal que deja pasar la luz a su interior.

Fuente (transistor): El terminal de un transistor del tipo Mosfet que se conecta a masa. Cuando el terminar de control o puerta de este transistor recibe una tensión, la fuente y el drenador se conectan, como si fuese un interrutpor, y de esta forma se cierra el circuito que está siendo controlado.

Fuente de alimentación: Una fuente de energía, generalmente un batería, un transformador, o también un puerto USB de un ordenador. Puede ser de varios tipos, estabilizada o no estabilizada, de tensión alterna o de tensión continua. Por lo general se especifica la tensión o tensiones junto con la máxima intensidad que puede proporcionar antes de que falle.

Función: Un bloque de código que ejecuta una tarea concreta cuando es llamada por el programa principal.

H

Hoja de datos: Un documentos escrito por ingenieros para otros ingenieros o técnicos en electricidad o electrónica que describe el diseño, los parámetros principales, las curvas características y los circuitos típicos de los componentes electrónicos. La información típica en la hoja de datos incluye la tensión máxima y la corriente que el componente puede soportar (valores máximos) así como también los valores típicos para su correcto funcionamiento. También se incluye un dibujo del encapsulado con las medidas y una explicación de la función de sus terminales.

I

IDE: Significa “Integrated Development Environment” o “Entorno de Desarrollo Integrado”. El IDE de Arduino es el lugar donde se escribe el programa que se va a cargar a la tarjeta Arduino. Contiene todas las funciones que Arduino puede “entender”. Otros entornos de programación como Processing tienen su propio IDE.

Index: El número suministrado en una matriz que indica a qué elemento se está refiriendo. Los ordenadores están indexados desde cero, lo cual significa que comienzan a contar desde 0 en lugar de 1. Por ejemplo, para acceder al tercer elemento de una matriz llamada tonos, se escribe tonos[2].

Inducción: El proceso de usar energía eléctrica para crear un campo magnético. Se usa en los motores para hacerlos girar.

Int: Un tipo de variable que puede guardar un número entre -32.768 y 32.767.

Interruptor: Un componente que puede abrir o cerrar un circuito eléctrico. Existen varios tipos de interruptores; los que se incluyen en el kit no son exactamente interruptores sino pulsadores, ya que solo cierran el circuito en el momento de presionarlos y solo si se mantienen presionados, si se deja de presionar el pulsador abre el circuito.

L

LED de cátodo cómun: Un tipo de LED que dispone de varios led en su interior para genera varios colores, con un solo cátodo y un ánodo por cada led interno (normalmente tres).

Ley de ohm: Una ecuación matemática que demuestra la relación entre resistencia, intensidad y tensión. Siendo esta ecuación V(tensión) = I (intensidad) X R (resistencia).

Librería: Una pieza de código que aumenta la funcionalidad de un programa. En el caso de las librerías de Arduino, se utilizan para activar la comunicación con un componente electrónico en particular (por ejemplo un display) o se usan para manipular datos.

Long: Un tipo de variable que puede guardar números muy grandes, de -2.147.483.648 a 2.147.483.647.

M

Masa: El punto de un circuito donde normalmente hay cero voltios y que se conecta a las carcasas metálicas de los equipos cuando es posible. Normalmente se suele considerar la masa como el negativo de la fuente de alimentación o de una pila, y sin este negativo o masa el circuito no se cierra y por tanto la energía eléctrica no puede fluir.

Matriz: Son un grupo de variables que se identifican con un único nombre dentro de un programa, y se accede a estas variables dentro de la matriz mediante un índice numérico o index.

Microcontrolador: El cerebro de Arduino, se trata de un circuito integrado que se puede programar para recibir, enviar, procesar y visualizar información.

Mili segundo: La milésima parte de un segundo 1/1000 segundos. Arduino ejecuta sus programas muy rápido al ejecutar muchas instrucciones cada mili segundo. Cuando se utiliza delay() u otras funciones que se basan en tiempo se indican en mili segundos. Por ejemplo, para introducir un retardo de 1 segundo se escribe delay(1000).

Modificador: Son palabras reservadas por el programa para modificar el comportamiento de las variables. Por ejemplo, la variable int PinSalida = 13; guarda un número entero que se puede modificar en el programa, en cambio si se usa con: const int PinSalida = 13; al igual que antes guarda un número entero (13) pero además se indica que se trata de una variable constante que no se puede modificar.

Modulación por Ancho de Pulso (PWM): Se trata de una forma de simular una salida analógica cuando se utiliza una salida digital. PWM hace que un terminal cambie su valor entre el estado bajo y el estado alto a un ritmo muy rápido. El tiempo que está en estado alto con respecto al tiempo que estado bajo determina el resultado de la simulación analógica.

Monitor serie: Un circuito que está incorporado dentro de la tarjeta de Arduino y que permite enviar y recibir información usando el IDE a dicha tarjeta. Ver las opciones de la función Serial() en este enlace: **arduino/serial**.

O

Objeto: Una copia de una librería. Cuando usando la librería Servo, se ha creado una copia llamada MiServo, MiServo sería el objeto.

Ohm: Unidad de medida de las resistencias. Representada por la letra omega (Ω).

Onda cuadrada: Un tipo de forma de onda que se identifica por tener solo dos estados, alto y bajo. Cuando se usa para generar tonos, los sonidos generados con esta onda puede resultar poco naturales.

Optoacoplador: También se conoce como un opto-aislador, foto-acoplador, foto-interruptor y opto-interruptor. Un diodo LED de infrarrojos es combinado con un foto transistor sensible a la luz infrarroja dentro de un encapsulado sellado. El LED está colocado de forma que ilumina al foto transistor cuando se le aplica una tensión, y de esta manera el foto transistor conduce al recibir la iluminación del LED. Se utiliza para proporcionar un alto grado de aislamiento entre el circuito de control del LED y el circuito de salida del foto transistor, además de no existir una conexión eléctrica entre estos dos circuitos.

P

Paralelo: Los componentes de un circuito que están conectados a través de los mismos dos puntos se dicen que están en paralelo. Los componentes conectados en paralelo tienen siempre la misma tensión entre ambos. Por ejemplo, una resistencia R1 en paralelo con una resistencia R2 tendrán la misma tensión entre sus extremos.

Parámetro: Al declarar una función se nombra un parámetro que sirve de puente entre las variables locales de la función, y los argumentos que recibe cuando se llama a esta función.

Periodo: Tiempo durante el cual se repite la misma forma de onda. Si se varía el periodo de una onda también se varía la frecuencia de la misma, ya que la frecuencia es igual a 1/Periodo. El periodo se mide en segundos y la frecuencia en ciclos por segundo o hercios.

Polarizado: Los terminales de los componentes polarizados (por ejemplo diodos LED, condensadores electrolíticos, transistores, etc) tienen diferentes funciones y hay que conectarlos de una forma determinada. Los componentes polarizados si no se conectan correctamente no van a funcionar, se podrían estropear e incluso dañar a otros componentes del circuito en donde se han montado mal. En los componentes no polarizados (por ejemplo las resistencias) sus terminales se pueden conectar de cualquier forma.

Processing: Un entorno de programación que se basa en el lenguaje java. Usado como una herramienta para introducir a todo aquel que quiera trabajar con Arduino en conceptos de programación, y en entornos de producción. El IDE de Arduino está escrito en Processing, así que le resultará muy familiar. Además Arduino y Processing comparten la misma filosofía y motivación, el poder crear herramientas de código abierto que permitan a personas que no son técnicos trabajar con hardware y software.

Pseudo código: Un puente entre la escritura en un lenguaje de programación y el uso de términos que el ser humano puede entender al leerlos.

Puerta (transistor): El terminal a través del cual se controla a un transistor del tipo Mosfet y que se conecta a una salida de Arduino. Cuando a la puerta del transistor se le aplica una tensión de +5V (estado alto), se comporta como un interruptor cerrado entre los terminales drenador y surtidor, cerrando de esta forma el circuito y haciendo que el componte conectado a él comience a funcionar (zumbador, bombilla, relé, etc).

R

Relación cíclica: Una relación que indica la cantidad de tiempo que una parte de una onda se encuentra en estado alto con respecto al periodo. Cuando se utiliza un valor de PWM de 127 (de un total de 256), se produce una relación cíclica en la onda que se genera del 50%, esto quiere decir que estará en estado alto el 50% del periodo.

Resistencia: La propiedad que presentan muchos materiales al paso de una corriente eléctrica a través de ellos y que sirve para indicar si es un conductor de la electricidad. La resistencia de un material se pueden calcular a partir de varios parámetros físicos del mismo, pero también se puede usar la ley de Ohm para calcular la resistencia en un circuito: R = V/I

S

Sensor: Componente que mide un tipo de energía (como luz o calor o energía mecánica) y la convierte en energía eléctrica, la cual Arduino puede procesar.

Serie: Se dice que los componentes están en serie cuando la corriente eléctrica fluye pasando por un primer componente y a continuación por el siguiente una vez a atravesado el primero, después la corriente pasa por el segundo componente para llegar al tercero y así sucesivamente. En los componentes en serie la corriente es la misma en todos ellos mientras que la tensión que aparecen en los extremos de cada uno de ellos es diferente.

Sin signo: Término que se utiliza para describir los datos que pueden guardar algunos tipos de variables, indicando que no puede guardar un número negativo. Es útil tener un número sin signo cuando es necesario contar en una sola dirección. Por ejemplo, cuando se hace un seguimiento del tiempo con la instrucción millis(), es aconsejable usar una variable que guarde un dato de tipo long (grande) sin signo.

Sketch: Término que se le da a los programas que han sido escritos dentro del IDE de Arduino.

Soldar: Proceso por el cual se realiza una conexión eléctrica al unir dos componentes electrónicos, un componente electrónico y un cable, dos cables o un componente electrónico que se une a una pista de cobre de una placa de circuito impreso. Para llevar a cabo este proceso se usar estaño, el cual se calienta a más de 200 grados para derretirlo y unir de esta forma a los materiales que se van a soldar. Esto proporciona un conexión sólida entre los componentes. Para soldar hay que usar una herramienta llamada soldador de estaño o estañador. Para realizar proyectos con las placas Arduino se recomienda usar un soldador de estaño de 30 vatios como máximo.

T

Tensión inversa: Tensión que aparece cuando una corriente atraviesa una bobina, la cual se opone a esta corriente que lo ha creado. Puede ser creada por los motores cuando giran y puede dañar el circuito de control del motor si no se protege mediante un diodo en paralelo con el motor y colocado en sentido inverso.

Transductor: Algo que convierte una forma de energía en otra forma de energía, como por ejemplo un altavoz, que convierte la energía eléctrica en energía acústica (sonidos).

Transistor: Dispositivo electrónico de tres terminales que puede funcionar como un interruptor o como un amplificador de señal. Una tensión o una corriente de control entre dos de sus terminales controlan una tensión o una corriente mucho más grande entre otros dos terminales diferentes (uno de ellos es común a la entrada de control y a la salida). Existen dos tipos de transistores, el Transistor Bipolar (en inglés BJT) y el Transistor de Efecto de Campo de Óxido Metálico (en inglés MOSFET). Los transistores bipolares se controlan por una corriente eléctrica aplicada al terminal de Base, y los transistores Mosfet mediante una tensión aplicada en su terminal de Puerta. Normalmente se utilizan para que Arduino pueda controlar cargas que consumen una intensidad mucho mayor que la que Arduino puede proporcionar (40 mA en un segundo), como puedan ser motores, relés o lámparas de incandescencia. Dependiendo de su construcción interna los transistores bipolares pueden ser de dos tipos, NPN o PNP, mientras que los Mosfet pueden ser de Canal N o de Canal P. Su construcción interna determina como se conectan en un circuito.

U

USB: Puerto de un ordenador que utiliza el estándar del Bus Universal en Serie (en inglés USB). Se trata de un puerto genérico que es un estándar en la mayoría de los ordenadores de hoy en día (2015). Con un cable USB conectado a un puerto USB de un ordenador el posible alimentar y programar una placa Arduino.

V

Variable: Un lugar de la memoria de un ordenador o de un microcontrolador en donde se almacena información para ser usada por un programa. Las variables almacenan valores los cuales pueden cambiar a medida que un programa se ejecuta. Las variables son de varios tipos según la clase de información que almacena y el tamaño máximo de esa información, por ejemplo, una variable de tipo byte solo puede guardar 256 valores diferentes, pero una variable del tipo int puede guardar 65.536 valores. Las variables pueden ser locales dentro un determinado bloque de código, o globales para ser usadas en cualquier parte del programa, (mirar Variable ocal y Variable global).

Variable global: Aquella a la cual se puede acceder desde cualquier parte de un programa. Se declara siempre antes de la función setup().

Variable local: Un tipo de variable que se usa durante un periodo corto de tiempo, para después olvidarla. Una variable declarada dentro de la función setup() de un programa sería local: después de ejecutar la función setup() Arduino deja de considerarla, es decir, no la tiene en cuenta. También son variable locales aquellas que se declaran dentro de funciones personalizadas por el usuario y que son llamadas desde el programa principal, olvidando Arduino estas variables cada vez que vuelve al programa principal.

Voltaje: Una medida de energía potencial, la cual produciría el movimiento de las cargas eléctricas si estuviese dentro de un circuito cerrado. Por ejemplo, una pila de 9V tiene un voltaje de 9 voltios, esto quiere decir que si se conecta una bombilla del mismo voltaje en paralelo con esta pila se producirá un movimiento de las cargas eléctricas (corriente eléctrica) que hará que la bombilla se encienda cuando las cargas la atraviesen. Al moverse las cargas en este circuito se produce un trabajo debido a la tensión o potencial de la pila. También se conoce con el nombre de tensión.

APUNTES Y LIBROS RECOMENADOS

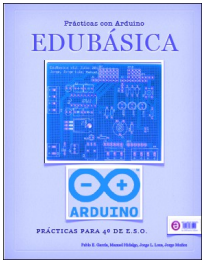
MANUALES Y APUNTES

Manual de introducción a Arduino tipo cómic por **Jody Culkin y traducido por Jose Manuel Escuder**. Se puede descargar desde:
http://playground.arduino.cc/uploads/Main/arduino_comic_es.pdf
Este manual de 15 páginas esta bajo una licencia Creative Commons By-NC-SA (Atribución – No Comercial – Compartgir igual 3.0).

Apuntes sobre Arduino Nivel Pardillo de **Daniel Gallardo García,** profesor de tecnología del IES Laguna de Tollón (Sevilla). Se puede descargar desde:
http://educacionadistancia.juntadeandalucia.es/profesorado/pluginfile.php/2882/mod_resource/content/1/Apuntes_ARDUINO_nivel_PARDILLO.pdf

Apuntes sobre Arduino Nivel Enteraillo de **Daniel Gallardo García,** profesor de tecnología del IES Laguna de Tollón (Sevilla). Se puede descargar desde:
http://educacionadistancia.juntadeandalucia.es/profesorado/pluginfile.php/2883/mod_resource/content/1/Apuntes_ARDUINO_Nivel_ENTERAILLO.pdf

Curso de supervivencia con Arduino de **Juan Gregorio Regalado Pacheco**
Descargar desde:
https://www.u-cursos.cl/uchile/2012/0/COMCAS/1/material_docente/bajar?id_material=700406
Este manual de 44 páginas esta bajo una licencia Creative Commons By-NC-SA (Atribución – No Comercial – Compartgir igual 3.0).

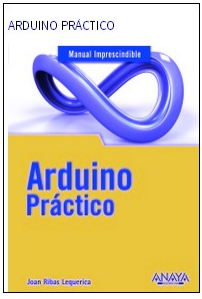


LIBROS

Libro sobre Arduino enfocado a la educación – Prácticas con Arduino 2 EDUBÁSICA por **Pablo E. García, Manuel Hidalgo, Jorge L. Loza, Jorge Muñoz.** Este libro es gratuito y se puede descargar desde esta página: http://www.practicasconarduino.com/

Arduino Práctico por **Joan Ribas Lequerica. Editorial** Anaya ISBN: 978-84-415-3419-3. Precio del libro 27.5 euros con iva. Enlace: http://www.anayamultimedia.es/libro.php?id=3273803

12 Proyectos Arduino + Android por **Simon Monk**. Editorial Estribor ISBN:978-84-940030-4-2. Precio del libro 25 euros. Enlace: http://www.superrobotica.com/S370530.htm



ARDUINO. Curso práctico de formación de **Oscar Torrente Artero**. Editorial RC libros. Precio 28 euros. Enlace: http://www.amazon.es/ARDUINO-pr%C3%A1ctico-formaci%C3%B3n-Torrente-Artero/dp/8494072501

Arduino le ayuda a hacer cosas que interaccionan con usted.
Ya sea un genio o un poeta, tenga diez o noventa años, queremos hacer posible que pueda construir grandes proyectos usando los ordenadores y la electrónica.

Las partes de este kit así como los proyectos que aquí se explican forma el esqueleto de sus proyectos. Arduino puede hacer que sus proyectos hagan algo. Depende de usted el hacerlos interesantes.